```
Trying 3106...Open

DIALOG INFORMATION SERVICES
PLEASE LOGON:
?
Logging in to Dialog

 ********
Maximum password attempts exceeded, try again later.
 ********
ENTER PASSWORD:
 ********
Welcome to DIALOG

Dialog level 00.05.02D

Last logoff:  26apr00 12:14:13
Logon file001 30may00 10:48:20
*** ANNOUNCEMENT ***
NEW FILE RELEASED
***New Scientist (File 369)
***Newsweek Fulltext (File 482)
***WIPO/PCT Patents Fulltext (File 349)

UPDATING RESUMED
***Bridge World Markets News (File 609,809)
***Fort Worth Star-Telegram (File 427)
***Federal News Service (File 660)
***Kansas City Star (File 147)

                            ***
RELOADED
***AIDSLINE (File 157)
***MEDLINE  (FIle 154,155)
***Books in Print (File 470)
***Kompass Latin America (File 586)

                            ***
REMOVED
***Global Mobility (File 64). Please use 2,6,8,63,65,94,99,108,238,266,
583.

>>>Get immediate news with Dialog's First Release
   news service.  First Release updates major newswire
   databases within 15 minutes of transmission over the
   wire.  First Release provides full Dialog searchability
   and full-text features.  To search First Release files in
   OneSearch simply BEGIN FIRST for coverage from Dialog's
   broad spectrum of news wires.

     >>> Enter BEGIN HOMEBASE for Dialog Announcements <<<
     >>>    of new databases, price changes, etc.      <<<
                  ****
KWIC is set to 50.
HILIGHT set on as '*'
***                                    ***

File   1:ERIC  1966-2000/May
       (c) format only 2000 The Dialog Corporation
```

```
*File   1: File has been reloaded. See HELP NEWS 1.

      Set  Items  Description
      ---  -----  -----------
? b 410
       30may00 10:48:27 User259891 Session D9.1
            $0.40    0.115 DialUnits File1
     $0.40  Estimated cost File1
     $0.01  TELNET
     $0.41  Estimated cost this search
     $0.41  Estimated total session cost   0.115 DialUnits

File 410:Chronolog(R)  1981-2000 Mar/Apr
       (c) 2000 The Dialog Corporation plc

      Set  Items  Description
      ---  -----  -----------
? set hi ;set hi
HILIGHT set on as ''
HILIGHT set on as ''
? b 15, 9,623, 810,
275,624,813,626,621,16,160,148,20,77,35,583,2,65,233,99,473,474,475,625,268,62
6,267,139
       30may00 10:50:37 User259891 Session D9.2
            $0.00    0.054 DialUnits File410
     $0.00  Estimated cost File410
     $0.60  TELNET
     $0.60  Estimated cost this search
     $1.01  Estimated total session cost   0.169 DialUnits

SYSTEM:OS  - DIALOG OneSearch
  File  15:ABI/INFORM(R)  1971-2000/May 26
         (c) 2000 Bell & Howell
  File   9:Business & Industry(R)  Jul/1994-2000/May 29
         (c) 2000 Resp. DB Svcs.
  File 623:Business Week  1985-2000/May W3
         (c) 2000 The McGraw-Hill Companies Inc
  File 810:Business Wire  1986-1999/Feb 28
         (c) 1999 Business Wire
  File 275:Gale Group Computer DB(TM)  1983-2000/May 30
         (c) 2000 The Gale Group
  File 624:McGraw-Hill Publications  1985-2000/May 25
         (c) 2000 McGraw-Hill Co. Inc
  File 813:PR Newswire  1987-1999/Apr 30
         (c) 1999 PR Newswire Association Inc
  File 626:Bond Buyer Full Text  1981-2000/May 26
         (c) 2000 Bond Buyer
  File 621:Gale Group New Prod.Annou.(R)  1985-2000/May 30
         (c) 2000 The Gale Group
  File  16:Gale Group PROMT(R)  1990-2000/May 30
         (c) 2000 The Gale Group
  File 160:Gale Group PROMT(R)  1972-1989
         (c) 1999 The Gale Group
  File 148:Gale Group Trade & Industry DB  1976-2000/May 30
         (c)2000 The Gale Group
  File  20:World Reporter  1997-2000/May 30
         (c) 2000 The Dialog Corporation plc
  File  77:Conference Papers Index  1973-2000/May
         (c) 2000 Cambridge Sci Abs
  File  35:DISSERTATION ABSTRACTS ONLINE  1861-1999/DEC
         (c) 2000 UMI
  File 583:Gale Group Globalbase(TM)  1986-2000/May 26
         (c) 2000 The Gale Group
  File   2:INSPEC  1969-2000/Apr W4
         (c) 2000 Institution of Electrical Engineers
```

```
  File  65:Inside Conferences  1993-2000/May W4
        (c) 2000 BLD[illegible]all rts. reserv.
  File 233:Internet & Personal Comp. Abs.  1981-2000/May
        (c) 2000 Info. Today Inc.
  File  99:Wilson Appl. Sci & Tech Abs  1983-2000/Apr
        (c) 2000 The HW Wilson Co.
  File 473:Financial Times Abstracts  1998-2000/May 26
        (c) 2000 The New York Times
  File 474:New York Times Abs  1969-2000/May 29
        (c) 2000 The New York Times
  File 475:Wall Street Journal Abs  1973-2000/May 26
        (c) 2000 The New York Times
  File 625:American Banker Publications  1981-2000/May 30
        (c) 2000 American Banker
  File 268:Banking Information Source  1981-2000/May W3
        (c) 2000 Bell & Howell
  File 267:Finance & Banking Newsletters  2000/May 26
        (c) 2000 The Dialog Corp.
  File 139:Econ. Lit. Index  1969-2000/May
        (c) 2000 American Economic Association

      Set  Items  Description
      ---  -----  -----------
? s double(w)dispatch
         1192096  DOUBLE
           67590  DISPATCH
      S1       5  DOUBLE(W)DISPATCH
? d s1/9/all
      Display 1/9/1     (Item 1 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

02207404     SUPPLIER NUMBER: 20942641    (THIS IS THE FULL TEXT)
We have mail.(Letter to the Editor)
Meyers, Scott; Tribble, David R.; Spangler, Dale; Cousins, David L.
C/C++ Users Journal, v16, n7, p100(3)
July, 1998
DOCUMENT TYPE: Letter to the Editor      ISSN: 1075-2838      LANGUAGE:
  English      RECORD TYPE: Fulltext
WORD COUNT:   2171    LINE COUNT:  00174

TEXT:
      Letters to the editor may be sent via email to cujed@mfi.com, or via
the postal service to Letters to the Editor, C/C++ Users Journal, 1601 W.
23rd St., Ste 200, Lawrence, KS 66046-2700.
     Dear CUJ,
      David Gould's article on double dispatching in the May CUJ was
interesting, but like many before him, David confuses the GOF Visitor

                                    -more-
?
      Display 1/9/1     (Item 1 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.
Pattern with the more general topic of double dispatching. As the latter is
a special interest of mine, I feel obliged to set the record straight.
      My fundamental concern is this sentence:
      "A celebrated solution to the problem of double dispatch
is the Visitor pattern..."
      Unfortunately, the Visitor pattern doesn't solve the problem of
double dispatch, it solves the problem of needing to be able to
add virtual-acting functions to a closed hierarchy. If you agree with the
informal definition of a pattern used by many in the patterns community, a
pattern is "a solution to a problem in a context." Technically speaking, I
guess there's no reason why one pattern can't solve more than one problem,
```

but people generally assume that each pattern solves a different problem. That being the case, it's important not to muddy the waters regarding which problem Visitor solves. If people come to believe that Visitor solves the double dispatching problem, they're less likely to remember that its original purpose was to solve a very different problem.

In his article summary, David states that "The Visitor pattern is an

-more-

?

Display 1/9/1 (Item 1 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

effective implementation of **double dispatch**." I believe this is precisely backwards. A particular manifestation of double dispatching (two virtual function calls) is used to implement the Visitor pattern. That is, Visitor is implemented via double dispatching, not vice versa. There are several other ways to implement double dispatching, and Visitor-like patterns could be implemented in terms of them, too, but Visitor itself could not. Visitor is, after all, "a solution to a problem in a context," so a different solution to the same problem would be, by definition, a different pattern. (Similarly, a set can be implemented via a hash table or via a linked list, but they are not the same design or data structure.)

Perhaps this all sounds like semantics gymnastics. I don't think it is. One of the most powerful aspects of patterns is that they raise the level of abstraction when talking about software designs. If somebody tells me they used pattern X, if I know what X is, I immediately know a lot about the problem that was being solved and the constraints affecting the potential solutions. So if somebody tells me they used Visitor, I should be able to assume they had a fixed hierarchy to which they needed to be able

-more-

?

Display 1/9/1 (Item 1 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

to add virtual functions. I'd probably also assume that they needed to be able to traverse an object structure while invoking these new virtual functions, because such traversals are how Visitor got its name in the first place.

But if some people say they use Visitor to solve the double dispatching problem, the pattern name becomes less precise, and I have to inquire about the specific problem being solved and the set of constraints in existence. I no longer know if there is a closed hierarchy or if traversal was part of the problem. My ability to communicate with the person describing their design is impeded. This is why terminological precision is important. (It is also why I nearly died when I learned that Win32 defines the term "critical section" in a way that's inconsistent with every standard historical use of the term. As a result, when a Windows programmer talks about a critical section, listeners familiar with the conventional definition must inquire about precisely what is meant. This is not progress.)

Readers interested in the general problem of double dispatching

-more-

?

Display 1/9/1 (Item 1 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

should know that the most extensive treatment of the topic for C++ can be found in David Chatterton's forthcoming doctoral thesis, "Dynamic Dispatch in Existing StronglyTyped Languages." A more accessible reference is Chatterton's and Conway's "Multiple Dispatch in C++ and Java," available at http://www.cs.monash.edu.au/~davidc/tools21.ps.gz. What I hope is a reasonably thorough treatment of the matter can be found in my More Effective C++: 35 New Ways to Improve Your Programs and Designs

(Addison-Wesley, 1996) pp. 228-251.

Readers interested in more detailed information on the Visitor Pattern and its variations may wish to consult the following references:

"Visiting Rights," John Vlissides, C++ Report, September 1995, pp. 12-16.

"The Trouble with Observer," John Vlissides, C++ Report September 1996, pp. 20-24.

"Acyclic Visitor," Robert Martin. Available at http://www.oma.com/PDF/acv.pdf.

"Visitors Revisited," Patrice Gautier, C++ Report, September 1996,

-more-

?

Display 1/9/1 (Item 1 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

pp.37-45.

"Default and Extrinsic Visitor," Martin Nordberg III, in Pattern Languages of Program Design 3 (PLOP 3) (AddisonWesley), 1998, ISBN 0-201-31011-2, pp. 105-123.

"Yet Another Visitor Pattern," Jack Reeves, C++ Report, March 1998.

Yes, that's a lot of references, but Visitor is an interesting pattern. However, it shouldn't be thought of as the same as double dispatching.

Scott Meyers

After rereading the chapter on Visitor in Gamma et al's Design Patterns, I must concur. It is an error to think of the Visitor pattern as an architecture that can be used to implement **double dispatch**.
That said, I think it is a pretty subtle error. It comes from confusing what is a very common implementation of Visitor with the Visitor pattern itself. This is actually very easy to do. For example, if you look at the chapter on Visitor in Design Patterns, you will find a big diagram of a class hierarchy (which happens to implement **double dispatch**),

-more-

?

Display 1/9/1 (Item 1 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

strategically labeled with the heading "Structure." Were I as naive as I was before reading your letter, I would have naturally assumed this diagram represented the "structure" of the Visitor pattern. It doesn't. It represents the structure of but one implementation of the Visitor pattern. (Sigh.) A picture may be worth 1000 words, but those 1000 words don't have to be right.

In further defense of David Gould, one of the things his article showed was how to add new GUI elements (buttons, boxes, and the like) to a fixed hierarchy of events, without modifying the event hierarchy. This is, in a sense, turning the Visitor pattern on its head: Visitor seeks to add new operations to a fixed hierarchy of objects. So Gould described his technique as an "inverted Visitor" pattern, in an attempt to convey what it did. The more correct "inverted **double dispatch**" would not have conveyed anything meaningful. We probably would have been in the clear if we had called it a "Visitor-like" pattern or an "Inverted Visitor-like" pattern. Thanks for enlightening us. -- mb

CUJ,

-more-

?

Display 1/9/1 (Item 1 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

In the "We have mail" section of the February 1998 issue of CUJ, Andre Oughton and Denton Karle both wrote about pointers and references. Oughton asked which of these is better:

void fool(int *myVal); void foo2(int &myVal);

The first requires the client to write an explicit & on the argument to fool, while foo2 doesn't.

Mr. Plauger responded by saying that he generally prefers to use an explicit pointer, and I agree. It's harder to mistake an explicit & for anything but pass-by-address, while it's not clear at all whether a function is being passed just a reference or a value since the syntax is the same for both.

On the other hand, many programmers feel that passing a const argument should look and feel like it's being passed by value. To wit, they prefer fooB over fooA:

void fooA(int myVal); void fooB(const int &myVal);

The syntax for calling fooA and fooB are identical. The difference is that fooA receives a copy of the argument you pass it, while fooB receives

-more-

?

a pointer (reference) to the argument. If myVal is a more complicated type, such as a class type, the difference can impact performance; the reference is faster. So sometimes using a const reference is okay.

Another thing I should point out is that references can be more dangerous than pointers. By its nature, once a reference has been established (seated) to an object, it can never be changed. Pointers, on the other hand, can be changed. This is extremely important if the object being referenced or pointed to is destructed. A pointer to a deleted object can be reset to NULL, but a reference to such an object can't be reset and becomes a dangling reference.

So programmers should choose carefully between pointers and references. I generally limit my use of reference variables to constructor and assignment operator arguments, and use pointers for everything else.

I stated that references can't be reseated; that's not entirely true. There is a loophole in the language that allows reference variables to be changed. The following code illustrates:

```
class Foo
```

-more-

?


```
{
public:
    Foo(int &i);
    void reseat_i(int &j);
private:
    int &   m_i;    // Reference var
};
Foo::Foo(int &i):   // Constructor
    m_i(i)          // Initializes m_i
{
}
void Foo::reseat_i(int &j)
{
    // In-place constructor,
    // reseats m_i
    new((void *)this) Foo(j);
}
```

-more-

?

(c) 2000 The Gale Group. All rts. reserv.

The reseat_i function uses a placement new call to invoke the constructor for Foo on the object pointed to by this, with the net effect of re-initializing (reseating) the m_i reference variable. Such programming tricks are valid, but I certainly cannot recommend them because of their inherent danger. This sort of thing makes reference variables even more dangerous because it breaks the implicit guarantee that references can't be reseated.

David R. Tribble Plano, TX dtribble@technologist.com http://www.flash.net/~dtribbleThanks for your insights. But I can't get too upset about a loophole in the language that requires so much conscious effort to exploit. I'm just about convinced that no language can be perfectly safe and consistent and still remain usable.

Well, the "pro-reference-argument" crowd has been strangely silent. I'd like to hear from someone who can argue (no pun intended)for using reference arguments instead of pointers. --mb

Dear Mr. Plauger,

This is just a short note to ask about the recently approved C++

-more-

?

Display 1/9/1 (Item 1 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

standard. I have seen a couple mentions in recent issues of CUJ that the standard is complete, but no mention of where it can be seen, etc. Is there a web site, or other place, where I could look at the standard?

Once upon a time there was a copy of one of the proposed Drafts posted on the Cygnus site. Has anyone done the same for the final version that everyone can look at?

Any information would be greatly appreciated. Dale Spangler derSpang@compuserve.com

Unfortunately, the Final Draft International Standard (FDIS) has so far been explicitly kept from easy public access. I just got a machine-readable copy myself a few days ago, so Pete Becket and I can review it before casting the Dinkumware ballot. I hear talk that the FDIS might soon become more freely available, but right now that is not the case.--pjp

Dear Sir,

I'm just a person who just wants to write simple programs. I have yet to find a book that gives me the kind of information I'm looking for. While

-more-

?

Display 1/9/1 (Item 1 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

the books I have looked at may have good information, it's not what I need:

Vb = Visual Basic d = Delphi 2.0 B3 = Borland Builder 3 vc = Visual C++

1. Edit box, string to float

```
vb  x = val(textl.text)
d   x := StrToFloat(Editl.Text);
b3  x = StrToFloat(Editl->Text);
vc  x = ?
```

Edit box, float to string

```
vb  Textl.Text = Format$(x,"###")
d   Editl.Text :=
      FloatToStrF(x, ffFixed, 7,2);
b3  Editl->Text =
      FloatToStrF(x, ffFixed, 7,2);
vc  ?
```

2. Radio Buttons

```
vb if option1.value = true
```

-more-

?
Display 1/9/1 (Item 1 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

```
            then c: "am"
       d  if checkboxl.state = cbchecked
            then c:= `am';
       b3 if(ChecBoxl->State == cbChecked)
            c="am";
       vc ?
```

And so on and so forth, with all the other visual controls. What's the syntax for getting information out of controls, putting information into the controls, controlling the controls, enabled, visible, hide, show, and so on and so forth?

I need some good books.

Sincerely,

David L. Cousins

Sounds like the first thing you need is to pick one programming language and stick with it. Then the syntax for accessing visual controls won't seem so alien to you. Visual Basic will probably get you up and running the quickest, but unless you "just want to write simple programs"

-more-

?
Display 1/9/1 (Item 1 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

for the rest of your life, I'd avoid it like the plague. It will seriously warp your sense of aesthetics. You could do a lot worse than Delphi, and the C->C++ route is still viable if you're more serious about programming than you let on. One language you didn't mention is Java, which is arguably just as easy as Visual Basic and a helluva lot prettier. Pick a language and then maybe we can talk about some books.--mb

FILE SEGMENT: CD File 275

- end of record -

?
Display 1/9/2 (Item 2 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

02165675 SUPPLIER NUMBER: 20527109

Double dispatch with an inverted visitor pattern. (Technology Tutorial)(Tutorial)

Gould, David

C/C++ Users Journal, v16, n5, p67(6)
May, 1998
DOCUMENT TYPE: Tutorial ISSN: 1075-2838 LANGUAGE: English
RECORD TYPE: Abstract

ABSTRACT: C++ has no inherent capability for implementing **double dispatch** techniques by using the single-dispatch virtual functions and overloading mechanisms simultaneously. Developers can nevertheless come close to true **double dispatch** by modeling a two-dimensional array of function pointers with a type of inheritance pattern called an inverted Visitor pattern. The Visitor pattern makes one of the table factors a Visitor and the other an Element of the object structure

-more-

?
Display 1/9/2 (Item 2 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)

corresponding to whatever has a true physical representation in the object. Using the Visitor pattern requires care because the Visitor class tree can grow rapidly while the Element class tree must remain fairly static. Inverting the pattern means moving Visitor-class event-handling algorithms to the Element classes. The compiler then resolves virtual functions and overloading at the same time.

DESCRIPTORS: Object-Oriented Programming; Software Architecture; C++ Programming Language; Programming Tutorial
FILE SEGMENT: CD File 275

- end of record -

?
Display 1/9/3 (Item 3 from file: 275)

02094280 SUPPLIER NUMBER: 19654833 (THIS IS THE FULL TEXT)
Dynamic design patterns in Objective-C: dynamic run times affect how programs are designed and built. . (Technology Tutorial)(Tutorial)
Grosso, William
Dr. Dobb's Journal, v22, n8, p38(10)
August, 1997
DOCUMENT TYPE: Tutorial ISSN: 1044-789X LANGUAGE: English
RECORD TYPE: Fulltext; Abstract
WORD COUNT: 3211 LINE COUNT: 00261

ABSTRACT: Language independence is a misleading concept. Patterns are seen as language independent, but choice of language limits the patterns that are available. For example, certain patterns are often used with Objective-C, but not with C++. Three standard design patterns, which are Visitor, Command, and Facade, are shown as implemented in Objective-C. Objective-C differs from C++ in that it uses the Smalltalk object model of

-more-

?
Display 1/9/3 (Item 3 from file: 275)

dynamic binding and also uses weak typing. The Visitor pattern is a common behavioral pattern, which could potentially be a dynamic pattern that supports code growth. The Command pattern encapsulates a request as an object. Both Command and Visitor patterns provide a way to break out parts of code and place them in shared libraries. The Facade pattern presents a method of structuring applications to take advantage of this flexibility.

TEXT:

Many software designers view patterns as a form of language-independent design. Pattern-Oriented Software Architectures: A System of Patterns, edited by Frank Buschmann (John Wiley & Sons, 1996), for instance, divides patterns in to three main groups--architectural patterns, design patterns, and idioms. Only the idioms (defined as "low-level patterns specific to a program language") are language dependent--the other patterns (and the implicit pattern language) rise above the level of programming language, much as the unified modeling language (UML) provides a common way to express designs.

-more-

?
Display 1/9/3 (Item 3 from file: 275)

This idea of language independence is seductive. It is also misleading. While patterns are language independent, language choice limits

the patterns that are possible, easily supported, and useful. Language also affects how applications are structured.

To illustrate, I'll examine some patterns often used with Objective-C but not usually used (or used differently) with C++. Both languages are object-oriented extensions of C. The main difference between them is that C++ has compile-time binding and fairly strong typing while Objective-C uses the Smalltalk object model of dynamic binding and weak typing. I'll start by examining two idioms commonly found in Objective-C programming. While simple, these idioms illustrate some of the key Objective-C programming techniques (although I use NextStep classes and frameworks to illustrate points, what I say also holds true for more generic Objective-C environments). I'll also examine how three standard design patterns--Visitor, Command, and Facade--are implemented in Objective-C.

Objective-C Programming Idioms

Idiom 1. Use Weakly Typed Delegation to Implement Secondary Roles of

-more-

?

Display 1/9/3 (Item 3 from file: 275)

Framework Objects. A framework is a set of objects designed to help users create a specific type of application or a specific part of an application. For most problem domains, there is a set of core abstractions that are independent of the framework. For example, consider a framework that helps you create a GUI. Almost every such framework includes an "Application" object. In NextStep and it is an Application, in Delphi's VCL it is TApplication, but the basic abstraction is the same. Similarly, all such frameworks contain some form of "Window," "Subview," "Button," and "Textfield." These framework objects are often the leaves on the framework's object graph.

Framework objects have a clearly defined role within any application. Application objects route events, and Window objects provide a place to draw. However, programmers often subclass objects to allow them to play a secondary role within an application. For example, Textfields are often subclassed within an application, giving them the ability to validate user input.

Standard object-oriented design principles suggest that secondary

-more-

?

Display 1/9/3 (Item 3 from file: 275)

roles be implemented via the Strategy pattern. Since frameworks are usually developed separately and implemented as shared libraries, using a strongly typed language results in Figure 1. However, Figure 1 can cause problems over the application life cycle. NextStep's Application object, for example, acquired nine new delegate methods between versions 2.0 and 3.3 of the framework. These were gradual modifications brought about by changes in how people use computers, as well as changes made to remedy oversights in the original design. In Figure 1, each of these changes would have required a recompilation of the application and a new release. Moreover, existing copies of the application installed on users' machines would need to be updated.

(FIGURE 1 ILLUSTRATION OMITTED)

One solution to this problem is to implement a strict versioning system. For example, the Windows machine I use has three distinct versions of MSVCRT**.DLL, and five versions of MFC**.DLL. If I try to launch my copy of Visual Eiffel, an alert dialog tells me I have the wrong version of CT13D32.DLL.

-more-

?

Display 1/9/3 (Item 3 from file: 275)

At best, strict versioning is a stop-gap solution. Depending on my applications, I might load all five versions of MFC**.DLL into memory, defeating the point of shared libraries. Additionally, older applications won't automatically inherit new functionality (or have bugs repaired)--they still use the older version of the framework. Moreover, minor updates to an application can become complex; if you want to use a feature found in the new version of the framework, you must update every object that depends on the framework (including all the strategy objects).

The Objective-C approach removes the abstract base class, giving the framework object a more complex setDelegate method. The framework object defines a set of strategic methods it will call if they are defined. Then, inside the setDelegate method, run-time type information (RTTI) determines which methods the delegate has implemented. If the delegate doesn't implement the strategic method, the framework object doesn't call it. This is illustrated in Listing One (listings begin on page 91), where the new delegate object is queried about each function. If the new delegate implements the function, the method pointer is obtained.

-more-

(PROGRAM LISTING ONE NOT REPRODUCIBLE IN ASCII)

This approach eliminates the versioning problem--older applications work with new frameworks. For example, most applications compiled under NextStep 2.0 can run using the shared libraries of NextStep 3.3. If the application needs to be updated to take advantage of a new feature in the framework, the necessary code change is minimal because only the object getting updated needs to be modified.

The essence of this idiom is the principle that the framework and application should evolve independently. It is not easy to see how to do this in a statically typed language.

Idiom 2: "Code" Is Nothing but "Resource" Misspelled. A typical application can display a large number of images, but rarely displays them all, or even a significant percentage of them. Because images are large, application developers frequently adopt a policy of lazy fetching, getting images from the disk only when they are about to be displayed. This lets you include a 600-KB GIF image in the "About" panel without forcing users to load a 600-KB image every time the application is launched.

-more-

In the era of sprawling programs, there are benefits to being able to do this with object code as well. Microsoft Word 7.0, for instance, includes functions to draw pictures, talk to databases, create forms, and perform sophisticated word-processing tasks such as mail merge. None of these features were used in writing this article. When I launch Word, however, it loads the object code for these functions--and Word subsequently takes longer to launch and consumes more resources when it is running.

A better approach would be to define a core program consisting of the tasks that are performed by most users each time they execute the application. When users try to use functions outside the core program, it would load a module containing code to perform the requested task.

Doing this properly requires that the loaded module and already-running code be dynamically linked, so that there exists only one object namespace between them. This way, the newly loaded code will be able to automatically locate Singleton objects, tie into existing event loops, and access existing objects with a minimum of programmer effort (objects

?

can be used as arguments without explicitly passing function pointers).

Fortunately, the Objective-C run time allows this functionality. You can message the run time to find out whether or not the required classes exist; if they don't, tell the run time to load them from a file and link them into the application. From then on, the loaded classes are first-class objects (exactly as if they had been part of the initial executable). This is illustrated in Listing Two, where the NXBundle class (provided by NeXT as a generalized "resource loader") loads in an obscure class.

(PROGRAM LISTING TWO NOT REPRODUCIBLE IN ASCII)

The Visitor Pattern

The Visitor pattern is one of the most commonly used behavioral patters. The idea is that an object represents a thing. It encapsulates attributes (facts about the thing) and methods (what the thing can do). However, objects frequently acquire extra methods. In particular, an object's methods often start to include "what can be done to the thing." The Visitor pattern is a common way of representing "what can be done to things" as separate objects. This lets you expand program functionality,

?

while keeping the basic object model intact.

A typical line of action for implementing the Visitor pattern is to create objects for your elements, container objects for your objects, and objects that encapsulate common complex operations on the data (see Figures 2 and 3). One thing that developers often overlook is that visitors can store cross-element states; for example, a visitor can create a report about the elements it has visited. This sort of operation must be external to the elements and is easily implemented as a visitor.

(Figures 2-3 ILLUSTRATION OMITTED)

Visitor has the potential of being a dynamic pattern that supports piecemeal code growth. As you discover operations that need to be supported, you add visitors that embody the operation and "fire them" at the appropriate data objects.

But the standard implementation of the Visitor pattern contains a flaw. Over the lifetime of a project, new visitors and elements (the objects that visitors act on) will be needed. Separate hierarchies for visitors and elements should enable painless modification of the software.

?

(Consider Figures 2 and 3.) Unfortunately, separate hierarchies are not easy to set up in the standard (strongly typed) version of Visitor.

Figure 2 implies that every concrete element must know about every visitor (to know whether to accept a particular visitor). And, of course, the visitor base class must know about every type of element (the visitConcreteElement methods must encompass every possible type of element). When there are many visitors and many elements, implementing and modifying the Visitor pattern becomes tedious and error prone. The overall effect is poorly understood and brittle software that has bugs.

This problem is partially caused by **double dispatch**: Elements accept (or reject) visitors, then call a method in the visitor and the ultimate function called is dependent on both the element and visitor. But this also means that elements are coupled to the visitors they accept.

Eliminating **double dispatch** simplifies the implementation.

If, however, **double dispatch** is necessary, you can still eliminate most of the compile-time dependencies in the code by having both the element's accept method and the visitor's operation use RTTI. In

-more-

?

Display 1/9/3 (Item 3 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

Listing Three, for example, the argument of the accept method is a pointer to a generic Visitor object. Inside the method, this argument is more carefully type checked (at run time) and certain types of visitors are accepted. This solution is similar to Robert Martin's Acyclic Visitor pattern in C++. The implementation of Visit is similar.

(PROGRAM LISTING THREE NOT REPRODUCIBLE IN ASCII)

Consider how this code needs to change when seven new elements are added. Obviously, seven new element classes need to be written and some visitors might have to be modified. But we have isolated the code changes to those visitors that are directly affected. Similarly, when a new visitor is added, only those elements that accept the visitor need to be modified.

Using RTTI localizes the code changes (to inside specific methods), eliminates compile-time dependencies, minimizes the number of objects that need. to be changed, and enables black-box reuse of most of the visitors and elements.

Object Serialization

As good object-oriented designers and developers, we actively practice

-more-

?

Display 1/9/3 (Item 3 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

information hiding (encapsulation). This means that at run time, a program is a huge graph of interconnected objects, each with their own private state. In addition, the connections between the objects are an important part of the application state. Object serialization is simply the idea that each object ought to be responsible for saving and restoring its own state and the connections it has (through instance variables). In other words, saving and restoring is best accomplished by traversing the object graph and invoking object-level save and restore operations.

This is a prime example of a simple implementation of the Visitor pattern. It also illustrates how loosely coupled the Visitor pattern can be in Objective-C, where there are three distinct types of objects: Serializers, Deserializers, and subclasses of Serializable. Both Serializers and Deserializers are visitors. Serializer visits each object in the object graph, getting a record of the object's state and saving the information out to a file. Deserializer takes a file and rebuilds an object graph from it. All the objects being serialized must inherit from Serializable.

-more-

?

Display 1/9/3 (Item 3 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

Serialization and deserialization are simple. In either Objective-C or C++, it follows the same basic communication pattern (see Figure 3 and Listing Five). But hidden in deserialization is a crucial question: Who creates the objects being instantiated?

(PROGRAM LISTING FIVE NOT REPRODUCIBLE IN ASCII)

In Objective-C, Deserializer is responsible for object creation. In the read Object method, Deserializer either reads in a reference to an already-created object or it reads in enough information to create the object. In the first case, it simply returns the preexisting object. In the

second, it creates the necessary object and calls an initialization method. This is code that does not need to change as the project grows.

Consider the C++ equivalent. Who creates the objects when deserialization occurs? Either Deserializer does (in which case Deserializer must know about each subclass of Serializable), or the subclasses of Serializable do by calling an initialization method (passing in the Deserializer). The second way eliminates code dependency, but has a potentially huge performance price (many extra objects will be created).

-more-

?

Display 1/9/3 (Item 3 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

The Command Pattern

The Command pattern (which encapsulates a request as an object) is one of the most fundamental patterns in object-oriented programming (see Figure 4). It is easy to understand and, aside from a slight decrease in performance, has almost no drawbacks when used properly. It lets you reduce code dependencies (the invoker object no longer depends on the receiver object) and it allows for run-time configuration changes (by changing the command object used by the invoker).

(Figure 4 ILLUSTRATION OMITTED)

Figure 5 illustrates the Command pattern with a Timer class--Object A creates an instance of ConcreteCommand and passes it to an instance of CommandTimer (which only depends on the abstract superclass Command).

(Figure 5 ILLUSTRATION OMITTED)

The standard implementation of Figure 4 in C++ uses templates. A template class is defined that takes the receiver object as a template argument. In addition, this class has a set method, which takes a pointer to a member function as its argument (in Listing Four, this information is

-more-

?

Display 1/9/3 (Item 3 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

passed in as part of the constructor). This template class is then used wherever a command object is needed.

(PROGRAM LISTING FOUR NOT REPRODUCIBLE IN ASCII)

Developers working in Objective-C use a slightly different solution: Instead of using pointers to member functions and strongly typed template arguments, they use selectors and pass in the receiver as an instance of type ID (see Listing Six).

(PROGRAM LISTING SIX NOT REPRODUCIBLE IN ASCII)

At first glance, these approaches seem almost equivalent. Indeed, the C++ approach has the added benefit of strong typing. However, the Objective-C approach--in particular, the use of selectors--has a significant advantage that is hinted at in Listing Six: You can easily convert a selector into a string (which is the method name). This string also can be converted back into the original selector.

Why is this an advantage? Consider object serialization and the related ways of making parts of the object graph persistent. The Objective-C approach to command objects is simply much more flexible and

-more-

?

Display 1/9/3 (Item 3 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

adaptable.

The Facade Pattern

You might have noticed a common thread in the patterns just discussed. Both of the idioms involve ways to break out pieces of code and put them in shared libraries or bundles. The design patterns show how to decouple

objects from each other for maximal flexibility. I'll now examine a commonly used architectural pattern that describes a way to structure applications to take advantage of this flexibility.

In the good old days of structured programming, saving an application's state was simple. Programs were divided into data and functions, and saving consisted of calling the function that wrote the data to some storage area. In object-oriented applications, things become a little more difficult. To the extent that you practice information hiding, object serialization is the natural approach to take.

Unfortunately, if m is the number of object references and n is the number of objects, serialization is O(mlog(n)). This is far too slow for serialization to be the saving mechanism in many applications.

-more-

?

Display 1/9/3 (Item 3 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)

But an Objective-C program that uses the four previous patterns will almost certainly use lots of facades as well. Notice in Figure 6 that Facades look a lot like cut points in the object graph. This leads to the natural Objective-C solution to the speed problem for object serialization: Serialize each facade to a separate Serializer (make each facade responsible for serializing the subsystem it abstracts).

(Figure 6 ILLUSTRATION OMITTED)

This can get complicated. If an object outside a subsystem bypasses a facade (and messages an object in a subsystem directly), then extra care must be taken during serialization (to avoid serializing objects to more than one location). And, deserializing (opening) becomes trickier as well--objects that bypass a facade will need to find objects within the subsystem. In practice, this comes down to making the facades used in serialization Singletons and making certain chat all connections to objects in the subsystem are mediated by the facade (so that, during deserialization, the connection can be restored).

Conclusion

-more-

?

Display 1/9/3 (Item 3 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)

The addition of a dynamic run time enables substantial changes in how a program is designed and built--changes that greatly ease your job over the application life cycle. Dynamic run times have another significant consequence, particularly when programs are designed using the aforementioned patterns. If you use bundles to encapsulate distinct pieces of object code, Facades to implement serialization, and loosely couple everything with Visitors and Commands (and ocher dynamic patterns that emerge in Objective-C programming), then you are, in essence, using a prototype-driven model of program design.

Acknowledgments

I'd like to thank Tom Lippincott and John Stytz for helpful comments on this article.

William currently works for Stanford University as part of the section on Medical Informatics. He can be reached at grosso@smi.stanford.edu.

SPECIAL FEATURES: chart; program; illustration

-more-

?

Display 1/9/3 (Item 3 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)

DESCRIPTORS: Programming Tutorial; C++ Programming Language

- end of record -

?
Display 1/9/4 (Item 4 from file: 275)

01911613 SUPPLIER NUMBER: 18046562 (THIS IS THE FULL TEXT)
The courier pattern.(Dr. Dobb's Sourcebook) (Technology Tutorial)(Tutorial)
Helm, Richard; Gamma, Erich
Dr. Dobb's Journal, v21, n13, p55(5)
Jan-Feb, 1996
DOCUMENT TYPE: Tutorial ISSN: 1044-789X LANGUAGE: English
RECORD TYPE: Fulltext; Abstract
WORD COUNT: 3042 LINE COUNT: 00248

ABSTRACT: A Courier pattern permits an object to pass arbitrary requests and data through a fixed interface. Information to be sent between objects must be packaged as an object itself, which must be sent as an argument to requests. The Courier pattern is useful in situations where a fixed interface may be inadequate for passing all of the required information to a recipient, or when requests sent between objects cannot be anticipated in advance. It may also be useful in situations where the class and interface

-more-

?
Display 1/9/4 (Item 4 from file: 275)

of the receiver are unknown to the sending class. Under the Courier model, few assumptions about the recipient's interface are required; the interface only has to be defined. Consequently, the sender and recipient can be loosely coupled. It also permits broadcasts to be executed without having to consider the class or interface of multiple receivers.

TEXT:
In previous columns, we've stressed the importance of ensuring that objects that send a request assume only that objects receiving those requests support a particular interface, but not anything about the receiving objects' concrete class (that is, its implementation). This ensures that senders are decoupled from receivers, and results in a characteristic design where the definition of operations refers only to abstract -- not concrete -- classes. Several design patterns from our book Design Patterns: Elements of Reusable Object-Oriented Software involve decoupling senders and receivers of requests; the Observer pattern discussed in our "Patterns and Software Design" column (Dr. Dobb's

-more-

?
Display 1/9/4 (Item 4 from file: 275)

Sourcebook, September/October 1995) is one of them.
As Figure 1 illustrates, the observer pattern decouples Subjects from their dependent Observers by defining an interface for signaling changes in the Subject. When decoupling senders and receivers, a common design issue is the kind of information passed between them, The simplest solution is to pass no information at all. Listing One implements a Notify() operation that doesn't pass any information. Since no information is passed to the observer and its Update operation, the Observer has to find out what changed by asking the Subject for its current state. In other words, it has to "pull" the state from the subject. When it is expensive for the Observer to determine what changed, the pull approach can be also be expensive. Clearly, there needs to be a way for the Subject to pass more-specific

change information. The challenge, then, is making a concrete subject pass specific information about what changed to its concrete observers, given that the interface between subjects and observers is defined by the abstract classes Subject and Observer.

One way to solve this is by adding a void* as a parameter to a tag

-more-

?

Display 1/9/4 (Item 4 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

that identifies the information passed in the void*. For example, you could change the Update operation of Observer to Update (longwhatChanged, void* info). This approach is not particularly type safe, and a void* in a higher-level class interface is not particularly elegant.

Chain of Responsibility is another pattern that decouples the sender from the receiver. It does this by passing a request along a chain of potential receivers. As Figure 2 suggests, event handlers are a good use of this pattern. With the Chain of Responsibility pattern, objects at the front of the chain try to handle requests (events) generated by some initial sender. if they cannot, they pass the request along to the next object in the chain, hoping that some object down the line will eventually be able to handle it. Chain of Responsibility requires that each object in the chain have an interface for handling the request. Again, the problem is, how can a sender pass arbitrary information to its candidate receivers, or how can new types of events be handled by the chain of objects? Defining a fixed interface for a fixed set of events such as HandleMouseEvent or HandleTimerEvent is not a viable solution when the set of events is

-more-

?

Display 1/9/4 (Item 4 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

extensible. For example, if the system were extended to support drag-and-drop events, you would have to change existing classes and add a HandleDropEvent operation.

The Mediator pattern also decouples objects by having them refer to each other indirectly through a Mediator object (see Figure 3). The Mediator object is responsible for routing requests between Colleague objects. In the Mediator pattern, Colleague objects do not send requests directly to each other but instead go through an intermediary Mediator. This nicely decouples Colleagues from each other, but also means that Colleagues can only communicate with each other via the fixed Mediator interface. Again, you have the problem of defining an interface that enables concrete colleagues to pass specific information to one another.

To summarize, decoupling objects by defining them in terms of interfaces defined in abstract classes is often the basis for reusable, object-oriented designs. But the objects may become so decoupled from each other that their interfaces do not allow for the passage of specific information. The Courier pattern is one solution to this problem.

-more-

?

Display 1/9/4 (Item 4 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

The Courier Pattern

Courier patterns allow objects to pass arbitrary requests and information through a fixed interface. The key to the Courier pattern is to package the information to be sent between objects as an object itself and pass this "message object" as an argument to requests.

Let's begin by exploring the Courier pattern in the context of the Observer pattern. Assume that a TextSubject class defines an editable text buffer, and a TextObserver somehow displays the TextSubject (Figure 1). The

TextObserver class receives notifications about changes in its TextSubject. Implementing a TextObserver using the interface defined for the Observer class in Figure 1 would require that, when the TextSubject changes in any way, the TextObserver has to retrieve the complete text buffer from the TextSubject. There is no way for the TextObserver to determine what changed in the TextSubject using just the Update() interface. Ideally, the TextSubject would be able to inform its Observer exactly what about itself changed. Using the Courier pattern, we would package this information into an object and send that to the TextObserver.

-more-

?

Display 1/9/4 (Item 4 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

We first define a message object using the Notification class. In this case, we define it as an empty class; see Listing Two. Then we change the Notify and Update interfaces of Subject and Observer; see Listing Three.

A TextSubject that wants to pass additional information to its Observers can now define a TextChangeNotification subclass that stores additional information about the changed text range. Assume in Listing Four that there is already a TextRange class that can be used to specify a range of changed text. Now the TextSubject can pass a TextChangeNotification to notify its observers; see Listing Five. Finally, a TextObserver can use this additional change information to optimize how it updates itself; see Listing Six.

Other kinds of notification requests (text deletion, style changes) could be easily added and handled in a similar way.

In the Courier pattern, notification requests are represented as objects. The recipient of the request must analyze the message object and handle it appropriately. First, the notification object has to be decoded to identify its concrete class (we did this using the run-time,

-more-

?

Display 1/9/4 (Item 4 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

type-identification mechanism provided by C++). Then the appropriate code to handle this request must be determined and executed. In some situations, if the recipient class cannot handle the request, it may have to defer it to its parent classes. This adds run-time overhead.

Participants

As Figure 4 shows, the key participants in the Courier pattern are:

* Message class (Notification in listing Six), which defines the message passed to a Recipient. It can represent requests and also define parameters and return values of the request. The TextCbangeNotification represented a text-change request and also carried the request parameters (the range of text that changed). The interface to Message must allow clients to determine the kind of message (this can also be done by language mechanism).

* Sender, which creates a concrete Message and sends it to a Recipient.

* Recipient class (Observer), which defines the interface to Handle messages; for example, Handle(Message&). The Recipient must decode the

-more-

?

Display 1/9/4 (Item 4 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

Message, determine which code to execute, and retrieve any parameters from the Message.

* ConcreteRecipients (TextObserver), which receives Messages and decodes them in the implementation of their Handle operation.

* ConcreteMessages (TextChangeNotification), which defines the actual contents of a particul[redacted]Message.

* Intermediary classes, which forward Messages from the initial Sender on to their ultimate Recipient.

Note that in Figure 4, only the Sender and the ConcreteRecipient know about the ConcreteMessage.

Applicability

When should you use the Courier pattern? The first situation is when a fixed interface defined between abstract classes is insufficient to pass all information required to recipients that are concrete subclasses of these abstract classes.

A second situation is when the requests to be sent between objects cannot be anticipated in advance, and you need to provide hooks to allow

-more-

?

the interface to be extended for new requests. This occurs in event handlers implemented using Chain of Responsibility when you want to extend the kinds of events the handlers can process.

In the third situation, the class and the interface of the receiver are unknown to classes sending requests, because the requests are sent via a third object. In Mediator and Chain of Responsibility, the class and even the interface of the class of object receiving requests are unknown to the sender. Rather than having the intervening object modifying or adapting the requests passing through it so it can be received by the ultimate recipient, we instead define a single Handle(Message&) interface to allow the Messages to be passed to the final Recipient unchanged.

Most uses of Courier are for more-loosely decoupling classes than is possible through interfaces defined by abstract classes. Another situation in which Courier can be very useful is during initial system prototyping and development.

One problem with statically typed languages (such as C++) is that you must explicitly specify the interfaces to abstract classes in their

-more-

?

declaration. You also have to duplicate much of this interface declaration (certainly for the virtual functions) in all subclasses inheriting from these abstract classes. Yet, as systems evolve, their interfaces change. This is especially true of newer systems, where it usually takes a few design iterations to find the major abstractions and specify their interfaces correctly. Changing interfaces to classes in C++ can be a costly operation. Not only do all subclass declarations and definitions need to be rewritten, but a significant amount of compile time may be spent in all clients implemented in terms of this interface.

For systems being prototyped or evolving rapidly, an alternative is to use the Courier pattern to send requests between classes in a system. Because Courier allows interfaces of classes to be changed just by modifying class implementations (or more precisely, the dispatching code in their Handle(Message&) operations), there is much less cost in changing interfaces. You simply have to add new Message subclasses and the appropriate dispatching code.

When the interfaces eventually begin to "harden," the implicit

-more-

?

dispatching provided by the Courier pattern can be replaced by direct requests to operations defined explicitly by the classes. The bodies of these operations will be what was previously the bodies of the If statements in the Recipient's dispatching operations.

Advantages and Disadvantages

The Courier pattern offers a number of benefits:

* Sending Messages to a Recipient class requires few assumptions about the Recipient's interface. It only has to define an interface to handle messages. This allows the Sender and Recipient to remain loosely coupled.

* It permits message-sending strategies such as broadcasting messages to all possible Recipients without caring about the class or interface of the receivers (only that they implement a Handle message). Receivers not interested in a particular message can simply ignore it. This avoids base classes becoming bloated with an interface that supports all possible operations. All a base class has to provide is a Handle(Message&) operation.

* Adding a new kind of message does not require changing existing

-more-

?

Display 1/9/4 (Item 4 from file: 275)

classes. Simply define a new Message class and implement the code that decodes and executes the message.

The Courier pattern also has some inherent disadvantages:

* The cost of Message dispatching. Decoding the Message to determine its class, retrieving any parameters, and finding the operation to perform must be done explicitly with conditional statements defined in the recipient. This is inelegant and can be less efficient than using the language's dispatching mechanisms directly.

* There is no guarantee that a Message can be handled by a Recipient. The compiler will not check that Messages are valid for a Recipient, that the message is decoded correctly, or that the operation to be performed by the Recipient is the correct one.

* The implementation of the Handle operation determines completely what a Recipient can do. It is not possible to infer any semantics about a Recipient from its interface.

Implementation Issues

One of the challenges when implementing Courier patterns is

-more-

?

Display 1/9/4 (Item 4 from file: 275)

identifying the Message classes. To be able to handle a Message, a Recipient must be able to identify the kind of Message it receives. The Message can define an accessor function that returns an identifier for the class as a constant or a string. This requires that the sender and receiver agree on the encoding. Alternatively, the receiver can use the language-supported, run-time type information, as we have in the Observer example. In this case, there is no need for special Message encoding conventions.

A second issue is acknowledging Message receipt. Senders sometimes need to know whether a Message was handled or not. For example, when propagating a message along a chain of responsibility, the sender can stop propagating the request once it is handled. To provide this kind of information, the Handle(Message&) operation can return a Boolean.

Another implementation issue involves handling requests, which requires identifying the message, then performing the appropriate action. Message handling is often distributed across a base recipient class and its subclasses. The base class can handle a more-general message, and the

-more-

?
Display 1/9/4 Item 4 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.
subclasses can handle more-specific messages. To enable this kind of message handling the derived class has to invoke the inherited Handle operation when it receives a message that it does not handle, as in Listing Seven.

The Recipient's message dispatch code is essentially boilerplate code. Where possible, the burden of having to create this boilerplate should be removed from the client. One way is to have a code generator (often called a "wizard") that generates the code based on some specification.

Self-Dispatching Messages

One potential problem with implementing Courier patterns is that all the dispatching code is in the Recipient. An alternative solution is to make Messages self dispatching.

When we perform the dispatching, we have different Recipients and different Messages. Code that actually gets executed when a message is received depends on both the concrete classes (more strictly speaking, the type) of both the Recipient and Message objects. There are languages (CLOS, for example) that support operations (multi-methods-methods) that can be

-more-

?
Display 1/9/4 (Item 4 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.
dispatched on more than one parameter type. In such languages, message dispatching would be handled by language directly and not implemented programmatically by the programmer. However, languages such as C++ and Smalltalk dispatch operations on the type of only one object -- the object receiving the request.

One technique to dispatch operations on multiple classes in such languages is "**double dispatch.**" We use a variation of this technique in Listing Eight. (The only implementation we're aware of that uses this technique is the Input System of the Taligent Frameworks.

The Messages dispatch themselves. To do so, the base Message class defines a Dispatch function that takes a Handler as a parameter. The Handler class itself has the interface in Listing Eight. The Message base class implements the Dispatch function by simply calling HandleMessage on the handler passed to it; see Listing Nine. A Sender uses these classes in Listing Ten. So far this appears not very useful, as we've only introduced an additional level of dispatching. But Sender need know nothing about the concrete classes of the Message or the MessageHandler.

-more-

?
Display 1/9/4 (Item 4 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

Next, we define abstract classes that specify an interface for classes that wish to handle particular kinds of messages. Let's take a TimerMessage class defined as in Listing Eleven. For a TimerMessage, we define the TimerMessageHandler class in Listing Twelve. These handler classes for specific messages are used as mixin classes. A class that wants to handle a TimerMessage has to inherit from the TimerMessageHandler class and implement the pure virtual HandleTimerMessage function; see Listing Thirteen. Since the Sender sees only Messages and MessageHandlers, you may wonder how a particular Message finds its way to the specific MessageHandler. We first request a message to dispatch itself to a particular handler, then the message requests the handler to handle it. Consider the implementation of Dispatch for TimerMessage. It uses run-time type information to check whether the handler is a TimerMessageHandler. If it is, the message is delivered as a TimerMessage by calling HandleTimerMessage; see Listing Fourteen. Dispatch invokes its inherited

operation that delivers the Message as an ordinary message. The handler can use the less efficient but more general, HandleMessage operation to handle

-more-

?

Display 1/9/4 (Item 4 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

the message.

The Sender's first dispatch shows that the message is in fact a TimerMessage. The message then works out the type of the handler to call the appropriate operation for it, in this case HandleTimerMessage.

This technique frees the client code from dispatching messages by delegating the dispatching logic to the Message itself. However, it does have some drawbacks:

* Each different message kind needs a corresponding Handler class, so the number of classes is increased by two for each new message kind.

* A class that handles many different message kinds has a HandlerClasses list.

* When a class wants to handle an additional message kind, its interface has to be changed, since an additional handler class has to be mixed in.

(PROGRAM LISTING OMITTED)

Richard is an architect with the IBM Consulting Group's Object Technology Practice in Sydney, Australia. Erich is an architect and object

-more-

?

Display 1/9/4 (Item 4 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)
(c) 2000 The Gale Group. All rts. reserv.

technologist at IFA in Zurich, Switzerland. They are coauthors of Design Patterns: Elements of Reusable Object-Oriented Software (Addison-Wesley, 1994).

SPECIAL FEATURES: illustration; program
DESCRIPTORS: Programming Tutorial; Object-Oriented Programming
FILE SEGMENT: CD File 275

- end of record -

?

Display 1/9/5 (Item 1 from file: 2)
DIALOG(R)File 2:INSPEC
(c) 2000 Institution of Electrical Engineers. All rts. reserv.

6074215 INSPEC Abstract Number: C9812-6110J-111
Title: Multiple dispatch: a new approach using templates and RTTI
Author(s): Pescio, C.
Journal: C++ Report vol.10, no.6 p.16-17, 20-4
Publisher: SIGS Publications,
Publication Date: June 1998 Country of Publication: USA
CODEN: CRPTE7 ISSN: 1040-6042
SICI: 1040-6042(199806)10:6L.16:MDAU;1-V
Material Identity Number: O697-98006
Language: English Document Type: Journal Paper (JP)
Treatment: Practical (P)

Abstract: Virtual functions allow polymorphism on a single argument; however, sometimes there is a need for multi-argument polymorphism. **Double dispatch** , commonly used in C++ to implement multi-methods, does not lend easily extensible code. Solutions based on function tables are difficult to implement and prevent repeated derivation.

-more-

?

```
      Display 1/9/5     (Item 1 from file: 2)
DIALOG(R)File   2:INSPEC
(c) 2000 Institution of Electrical Engineers. All rts. reserv.
The article focuses on two new techniques based on templates and Run Time
Type Identification (RTTI). The first is faster but less fieldable; the
second is slower, but allows new classes to be added without the need to
change existing ones. (6 Refs)
  Descriptors: abstract data types; C language; object-oriented languages;
object-oriented programming; software reusability
  Identifiers: statically typed language; multiple dispatch; templates;
RTTI; virtual functions; double dispatch; multi-argument
polymorphism; C++; extensible code; function tables; Run Time Type
Identification
  Class Codes: C6110J (Object-oriented programming); C6140D (High level
languages); C6120 (File organisation); C6110B (Software engineering
techniques)
  Copyright 1998, IEE

                                - end of display -

?
>>>Page beyond end of display invalid
?
?
? s financial (w) instrument
Processed  10 of  27 files ...
Processing
Completed processing all files
         5964382  FINANCIAL
          424319  INSTRUMENT
      S2    4252  FINANCIAL (W) INSTRUMENT
?
? s s2 and (object (w) oriented)
            4252  S2
          486221  OBJECT
          788993  ORIENTED
          152825  OBJECT(W)ORIENTED
      S3      37  S2 AND (OBJECT (W) ORIENTED)
? s s3 and PY<=1998
Processing
Processed  10 of  27 files ...
Processing
>>>One or more prefixes are unsupported
>>>  or undefined in one or more files.
Processing
Processing
Processed  20 of  27 files ...
Completed processing all files
              37  S3
        43764885  PY<=1998
      S4      34  S3 AND PY<=1998
? type s4/9/all

 4/9/1     (Item 1 from file: 15)
DIALOG(R)File  15:ABI/INFORM(R)
(c) 2000 Bell & Howell. All rts. reserv.

01505705  01-56693
Financial CAD's plug and play platform
Webb, Andy
Wall Street & Technology  v15n10  PP: 76-82  Oct 1997  ISSN: 1060-989X
JRNL CODE: WSC
DOC TYPE: Journal article  LANGUAGE: English   LENGTH: 4 Pages
WORD COUNT: 2441

ABSTRACT:  Armed with an aggressive pricing strategy, FinancialCAD Corp. is
```

rolling out a sin[redacted] server platform aimed at he[redacted]ng dealing rooms to define new instruments and rapidly build integrated financial systems. The new product, called FinancialCAD Server, set for release in October 1997, is designed to allow users to instantly create and share models of any type of **financial instrument** without needing any programming knowledge. By using a single server platform to manage all analytical and transaction data, the Server will also calculate valuation, risk and cash flow data for those models. Apart from FinancialCAD, other players in the field include Integral Development Corp., VardePappersData and Oy Trema. A unique feature of FinancialCAD Server is that it has been implemented in a 3rd-generation, 3-tier, client/server architecture. While FinancialCAD Server can service the most sophisticated users and their extensive customization requirements, it will also be available in a format that will allow smaller clients to get up and running rapidly.

TEXT: Headnote:

Armed with an agressive pricing strategy, Financial CAD is rolling out a single server platform aimed at helping dealing rooms to define new instruments and rapidly build integrated financial systems.

The release of FinancialCAD Corp's (formerly GlasscoPark) new Server must rank as one of the biggest shifts in territory seen to date in the financial IT market. Based in Surrey, British Columbia, the Canadian software company, known for its work with financial derivatives, has previously specialized in producing pricing models, with its current library extending to some 140 items. These were originally produced as Excel add-ins, but have more recently also become available as Visual Basic and C++ run time libraries for those developing inhouse applications. The company has built a strong reputation in this field for both quality and extremely aggressive pricing, but is now up against a very different breed of competitor, albeit in a surprisingly small market.

The new product, called Financial CAD Server, set for release in October, is designed to allow users to instantly create and share models of any type of **financial instrument** without needing any programming knowledge. By using a single server platform, which hinges on the Unified & Trade; Data and Object Model, to manage all analytical and transaction data, the Server will also calculate valuation, risk and cash flow data for those models. The company claims that clients will be able to develop and deploy scaleable, reliable and powerful applications that will easily integrate with existing systems.

But, defining the rather nebulous "financial server" market that FinancialCAD has just joined isn't that easy. While there are plenty of companies that offer toolkits to develop new financial applications, there are relatively few that can offer an automated **object-oriented** environment suitable for the rapid development and deployment of fully integrated financial solutions. Of these firms, Infinity Financial Technology Inc., based in Mountain View, Calif., has (by IT standards) attained almost grandfather status. Apart from FinancialCAD, other players in the field include Integral Development Corp. (founded by Harpal Sandhu on leaving Infinity) and two Nordic companies VardePappersData and Oy Trema.

While the attractions of a structured **object oriented** environment for rapidly developing new financial applications have been the topic of much brave talk in the past, the reality has typically been somewhat in arrears. However, there are at last signs that the technology is fast approaching a feasible and practical maturity. Deals are being done. Integral announced two major contracts this year: JP Morgan completed the development of a front-to-back office emerging markets derivatives application in six months using Integral's Universal Financial Server (UFS). And Bayerische Vereinsbank has taken delivery of UFS as the basis for an exotics system that will integrate to mid and back offices.

(Illustration Omitted)

Captioned as: FinancialCAD Server, set for release in October, will allow users to create and share models of any **financial instrument** without needing any programming knowledge.

Meridien Research, the Boston-based technology research firm, has recently produced a major report on the "financial server" market ("Development of Risk Management and Trading Applications using Object Technology"). Octavio Marenzi, the research director who compiled the report, feels that the timing of the technology's coming of age is certainly propitious. "The [application] development cycle has simply been too long and as new financial instruments have appeared the real bottleneck has been IT. In an attempt to circumvent this, people have developed products using manual processes such as spreadsheets, which is not a very satisfactory approach when it comes to maintaining controls or integrating with.other systems."

The financial server approach isn't a universal panacea, though. Some developers, who have made speed of application development the ultimate priority, can create problems for clients further down the line. "While they may get applications out quickly they lack the quality of data structure necessary for running meaningful data queries at a later date," says Marenzi. "With their approach, a new exotic structure isn't stored in the database in a structured format using a data model, but thrown in as a large binary object. If you wish to use this data for other applications (such as risk management) it cannot be accessed using SQL."

Financial server technology is also not necessarily the most appropriate solution for all. "It can be great for larger firms that have the IT resources to get the most out of it," says David Little, president, Derivate Services Corp. "For smaller firms the training aspects can prove too costly and they tend to do better buying off the shelf solutions."

Though the off-the-shelf approach may traditionally be preferred by second and third tier players, there are signs that financial server technology could soon be trickling down to them. The development of proprietary instrument models may not be feasible for them, but the developers of financial server technology have already defined most commonly (and some uncommonly) traded instruments. For the smallest participants, there is now no reason why this technology should not be available to them on a bureau, or even Internet, distributed basis.

DESKS CAN DEFINE INSTRUMENTS

The fact that this type of server approach allows the trader (rather than a technologist) to create and define financial instruments can also make for more effective data maintenance. Though approaches to this issue vary slightly among the vendors, the general rule is that the creator of a **financial instrument** on the system is also responsible for its maintenance and that of any associated data. As David Glassco, CEO of FinancialCAD points out: "Who better to maintain the FX spot data than the spot FX desk?"

Nevertheless, some see problems in allowing new financial instruments to be defined in the front office. "This approach has the potential to get very messy," says Marenzi. "No matter how well these things are constructed, there's always the possibility that the trader will do something silly and produce some nonsensical **financial instrument** that will have to be disinterred from the database and rectified. It's also very difficult to optimize this kind of approach. If you have a large database with several thousand transactions an hour passing through it, you really need to get into the code in order to do that, which is something that the trader is unlikely to be able to do."

UNIFIED DATA MODEL

The FinancialCAD Server has had a lengthy gestation, starting life as FEO (Financial Engineering Objects) in mid 1991. ("Until we discovered that FEO meant "ugly" in Spanish," says Glassco.) "We quickly learnt that we needed a normalized data model that managed the primary data relationships," says Glassco. "We also realized that we needed normalized analytical interrelationships, so that if (for example) a 90-day LIBOR rate changed it would automatically propagate throughout the entire system across FX, bonds, commodities, equities and related derivatives like swaps and options."

The unified data model in FinancialCAD Server also means that it is extensible. New types of financial instruments can be added and integrated into the model on an ongoing basis. It also provides for unification across an enterprise, so that (for example) the FX team in an institution is using the same curves as the bond team. At the same time, the unified model dramatically reduces the total number of quotes coming into the system by automatically selecting only those quotes that are actually needed to build such things as yield curves.

Apart from the unified data model, the other thing that Glassco cites as a unique feature of FinancialCAD Server is that it has been implemented in a third generation, three tier client/server architecture. "This is something that the financial sector hasn't recognized yet," he says. "Because the Server is based on a distributed **object oriented** framework, companies wishing to scale from a single user system to several thousand and deploy financial objects on multiple machines on multiple sites can do so."

Predictably Glassco is upbeat about the prospects for this technology, but despite the ability to define any **financial instrument** in FinancialCAD Server, he remains pragmatic about the likely speed of take up in the financial industry. "It takes a lot of time and energy for companies to re-engineer their thinking and systems. However, our view of the financial market is that eventually it's going to be a Visalike transaction pumping network, with multiple distributed components and transactions made across the Internet. We've built Server to play a part in that."

AGGRESSIVE PRICING

(Photograph Omitted)

Captioned as: "It all depends on how you ... define 'support'," says Integral's Raj Patel. "If support Includes workflow, security, audit, and full front to back office processing, I don't think anyone can claim to support all financial instruments. "

(Photograph Omitted)

Captioned as: "Our view of the financial market is that eventually it's going to be a Visa-like transaction pumping network, with multiple distributed components and transactions being made across the Internet," says David Glassco of FinancialCAD.

The pricing of FinancialCAD Server is aggressive to the point of violence, and therein could lie a problem. (A purchase price of $15,000, with an annual license fee of $5000 and $100 per month per user.) There is the risk that major investment banks will refuse to take a product seriously that is priced at this low a level, when they are more accustomed to price tags on this sort of product starting at around $250,000.

However, it is clear that the company is aiming at a much broader market. Glassco is unrepentant about this. "We already service some 800 corporate treasuries and want to price the product so that they can all afford it. We

surveyed our smaller customers when fixing the price to find what they could afford. Our costs aren't so high that we have to price it at $500,000 and if we do that we immediately restrict ourselves to the top tier banks."

Microsoft, whose treasury department is going to be the first beta site for the Server (and with whom FinancialCAD has close technology links), endorses this pricing ethos. (Though Financial CAD Server will run on any ODBC database, the foundations of the product are pure Microsoft - DCOM, NT, SQL and Transaction Servers) "You want the exposure and the continual usage of the product," says David GumpertHersh, Microsoft product and technology manager worldwide capital markets, in Redmond. "FinancialCAD wants the Server anywhere there's a machine that can use it - even in a three-person operation, They are aiming at higher volumes at a lower price with the broadest possible user base."
Gumpert-Hersh is also upbeat about the prospects for the FinancialCAD Server. "Derivatives are getting more and more intertwined with the everyday operations of treasuries. As a result corporate customers are becoming more sophisticated and making greater demands on their banks. Based on that, you're going to have a lot more value-atrisk calculations demanded from directors and board members. The real value of this product is the new architecture, which drives the product to a much higher level. If it can deliver what FinancialCAD already delivers in an add-in form it should be a real winner."

Though FinancialCAD Server has the ability to service the most sophisticated users and their extensive customization requirements, it will also be available in a format that will allow smaller clients to get up and running rapidly. The product will initially ship on pre-built servers - typically high-end Pentiums with 64MB of RAM. These will have all the FinancialCAD software (and the supporting Microsoft products, which includes SQL, NT and Transaction Servers) fully configured and tested. All clients will have to do is unpack the crate and plug the machine into their network. They will also have the option of having FinancialCAD set the machines and software up to their specific requirements.

(Chart Omitted)
Captioned as: Infinity's new FinEx application allows traders to model new financial instruments in an Excel spreadsheet which is fully integrated with the Infinity trading and risk management applications for control and audit purposes.

Other players in the field have not been inactive either. Infinity launched their FinEx product at the SIA Conference in June. FinEx, when used in conjunction with Infinity's integrated trading and risk management system, afford traders and other users the same opportunity to define new financial instruments in a spreadsheet automatically in a broadly similar fashion to FinancialCAD and Integral.

Infinity's senior product marketing manager Richard Walker is optimistic not only about FinEx but also about the validity of this approach in general. "Platform-based development has been gaining increasingly broader market acceptance. Infinity customers have demonstrated success with the Infinity Platform for some time. Now other participants are hoping to duplicate the platform approach, and have accepted that this technology is more substantial than a flash in the pan. Infinity's approach differs from Integral and FinancialCAD in that our FinEx offering is wholly integrated with the Infinity integrated trading and risk management system, which includes the Infinity Data Model and Fin++ Class Library. "
Like Glassco, Walker sees the technology developing a broader user base. "We've been very successful in selling the Infinity Platform into organizations with staff capable of developing C++ applications," he says. "However, the majority of end users of those applications have a different skill set - they're not system developers. What they are good at is using spreadsheet applications to model the financial behavior of the instruments they handle. By having access to the underlying analytics of Fin++ via an

Excel interface with which they are comfortable, these end users are empowered to develop [redacted]able financial tools very qui[redacted]y."

Integral's customers to date have used its Universal Finance Server to replace their existing financial technology infrastructure in building derivatives trading and treasury-wide risk management systems. Unlike Infinity and FinancialCAD, Raj Patel, managing director of Integral Europe, is skeptical about the "one vendor supports all financial products" approach. "It all depends on how you chose to define 'support'. If support includes workflow, security, audit, and full front to back office processing, I don't think anyone can claim to support all financial instruments. Also, the claim that a single data model can support every type of **financial instrument** from a trading and back office perspective and at the same time act as a risk management warehouse is simply not feasible," says Patel. "In the end, transaction data models and risk management data models should be designed to support their respective requirements," he says. Nevertheless, he does believe "the next generation of technology, namely the financial server, will eventually span the entire range of financial instruments primarily because the vendor merely provides a framework and the user has the ability to fill in the workflow pieces. Net result - a complete solution, to the user's specs, in record time."

COMPANY NAMES:
FinancialCAD Corp
GEOGRAPHIC NAMES: US

DESCRIPTORS: Software reviews; Electronic trading; Systems integration; Servers; CAD
CLASSIFICATION CODES: 9120 (CN=Product specific); 3400 (CN=Investment analysis); 5240 (CN=Software & systems); 9190 (CN=United States)

4/9/2 (Item 2 from file: 15)
DIALOG(R)File 15:ABI/INFORM(R)

01092308 97-41702
The promise and the cost of object technology: A five-year forecast
Pancake, Cherri M
Communications of the ACM v38n10 PP: 32-49 Oct 1995 ISSN: 0001-0782
JRNL CODE: ACM
DOC TYPE: Journal article LANGUAGE: English LENGTH: 18 Pages
SPECIAL FEATURE: Charts References
WORD COUNT: 13886

ABSTRACT: The Association for Computing Machinery (ACM) convened an Industry Advisory Board (IAB) sponsored by IBM's Object Strategy and Implementation Group, to discuss future applications of OT in industrial settings. The panel included leading experts from industry and academia representing the areas of **object-oriented** languages, tools, standardization efforts, current application areas, and predicted areas of future growth. Panelists were asked to look ahead to what might be expected of object systems 5 to 10 years in the future. A brief summary is given of the characteristics of OT that make it promising for widespread use in industry, followed by a discussion of the panelists' explanation of why OT has not yet been widely adopted in the business environment. An outline is presented of the panel's ideas on where OT research efforts might best be allocated in order to maximize the benefits for future industrial users of OT.

TEXT: **Object-oriented** techniques for analysis, design, and construction of software systems have been espoused with fervor in the computing literature. Advocates cite successes in a wide variety of applications, from graphical user interfaces and distributed databases to

large-scale financial simulations. The increasing number of books and articles published on topics related to object technology (OT) lends credence to the fact that this discipline has come of age. However, a number of pertinent issues have not been firmly addressed. To what extent will OT really affect business software and business processes? If OT is mature, why hasn't it been adopted as widely as many predicted? In what areas does OT offer the most promise, and as OT expands into these areas, what new challenges lie ahead?

To answer these questions, ACM convened an Industry Advisory Board (IAB) sponsored by IBM's Object Strategy and Implementation Group, to discuss future applications of OT in industrial settings. The panel included leading experts from industry and academia representing the areas of **object-oriented** languages, tools, standardization efforts, current application areas, and predicted areas of future growth. Panelists were asked to look ahead to what we might expect of object systems 5 to 10 years in the future.

Why OT is Important for Industrial Applications

OT offers several advantages that map well to recent changes in the industrial marketplace. As priorities shift to emphasize speed and responsiveness to market conditions, traditional software engineering methodologies like the so-called "water fall model" fall short. Lorette Cameron summarized the situation for many businesses: "We are being asked to do things much more quickly and we cannot plan it three years out. The solution that comes to mind immediately is just to grab a fourth generation language or some kind of tool so we will not have to go through that long cycle. We have solved a lot of short-term problems using very simplistic tools that have gotten the job done very quickly. The analogy that I use, however, is that we have fed our users a diet of junk food. It satisfies their hunger for the moment and we crank out applications very quickly. But then two years from now, these applications have no nutritive value."

Object orientation (OO), on the other hand, makes it possible to model systems that are very close in structure to their real-world analogues. The objective of OO design is to identify accurately the principal roles in an organization or process, assign responsibilities to each of those roles, and define the circumstances under I which roles interact with one another. Each role is encapsulated in the form of an object. This approach is quite different from more traditional analysis methods, whose emphasis is on process. Where a process oriented model stresses the sequencing of activities to accommodate chronological dependencies, a role-oriented model is concerned with the policies or conditions that constrain task performance.

Consequently, OO systems are remarkably responsive. As David Ungar pointed out: "Being able to create in the computer a self-consistent, understandable universe of things that matches the way you want to think about your problem gives you a lot of leverage in getting the computer to make these things dance the way you need them to, in order to make money off them." Flexibility and agility are the terms that are used frequently to describe the effects of applying OO to business engineering or software design. When objects are used to model the roles that contribute to an enterprise, they localize where changes are needed in the same way that a human manager takes responsibility for localizing which subordinate should respond to a situational change. It becomes possible to make a small, quick modification to a single object, with the result that the entire system is influenced in a very dramatic way.

In some ways, OT is an extension of the layered approach that is already familiar to software developers. Read Fleming maintained that in most current companies, "If you went down and looked at how the big IS systems work ... They've probably got things structured so there's a layer of data, and then there's a narrow layer of routines or applications that actually go and do the manipulations--they isolate the end user applications from

the data. Going to object technology is a way of making that structure explicit, rather than relying on convention. It helps you with managing it, because you understand it better." This advantage derives from the nature of the OO analysis process. The discipline of identifying key roles and interactions forces us to re-examine procedures and policies that might have been taken for granted. In many cases, it turns out that project management techniques are not as effective as they were assumed to be. As Alan Nugent commented: "Object technology brings these problems to light. For that reason alone, I think it's worth embracing on the part of large-scale software development efforts."

The benefits of OT are not limited to software, but are now being applied to business processes as well. A new field-**Object-Oriented** Business Engineering--has developed, replete with methodologies for using object systems as the basis for business re-engineering [9]. Ray Cline described the concept: "You have to engineer not just your software, but your business and human practices simultaneously. Historically we've had information systems that are trying to retrofit into the business model, or business models that are trying to retrofit with what the information systems can do. What we have here is the opportunity to build a new way of doing business that includes not only the bits and the bytes but the people and the processes." The roles, responsibilities, and interactions in an object system have clear analogues in a business architecture. Using OT to model an enterprise makes it possible to capitalize on over a decade's research in developmental methodologies, without imposing an artificial or unnatural structure.

The role-oriented nature of OT also makes it possible to restructure the information hierarchy so that it is less rigidly compartmentalized. Current dependence on layered or monolithic applications means that data is available only to certain individuals or under certain sets of circumstances. Restructuring that information as an object system means that data can become available in response to more diverse factors. Rick Stevens described how this can have a positive influence on business decisions: "Organizations today are trying to improve their internal infrastructure and to adopt more galactic information systems. I've been amazed when people ... were suddenly able to make contributions they wouldn't have been able to make had they not had that new access to information."

The Nature and Value of Software Components

The most widely cited advantage of OT is the fact that objects can be used as software components. Objects embody units of functionality that can be adopted by other programmers, both within the developer's organization and at the broader community level. The independent, modular nature of objects makes them ideally suited for reuse in other applications, without modification. At the same time, the ability to define subclasses means that the features of an object can be revised or added to with relative ease. The concept of software components has great appeal. As one author notes: "We still dream of 'software factories' which will cheaply produce high quality software" [6]. Advocates envision collections of software building blocks, available to the entire computing community, that will encourage open, cooperative software development (see [6] for a description of the techniques that would make this possible).

The notion of software recycling may have little immediate impact on industrial users of OT, however. Reusability does not come without a price tag. Typically, it is only after an OO application is complete that the developers understand which objects might have broader use. Those objects then must be restructured through a process known as generalization, that in turn may require revisiting several earlier stages of object design (see 8] for a survey of research on the life cycle of OO software).

Clearly, a key aspect of generalization is the specification of the

object's interface, which establishes how it will interact with other objects. This is not an easy thing to do. Ungar cautioned that "the notion of interface is much more of an illusion than people give it credit for. The problem is that nobody really knows how to formally write down the definition of an interface in a way that can be checked and will guarantee the thing will really work when used by different clients .... You try to get all your design decisions to hide behind interfaces so that changes percolate through as few levels as possible. But every honest programmer will admit there are times when you have to change your mind, and the interface changes." The problem of how to manage such changes has not been solved, particularly if components are in use across a number of organizations or if the components have been modified or built upon in any significant way.

It is also unlikely it will be possible to build complete applications from prepackaged components, at least in the next few years. In a few key areas, such as graphical user interfaces and distributed operating systems, object collections are already available. They typically provide only the most rudimentary functionality, however, so building a complex application still involves a great deal of new programming effort. For business applications, there are very few existing component libraries. As Cameron told the group: "I would just as soon never have to write another set of software to worry about the days when the New York Stock Exchange is open, and how they're different from when the London Exchange is open. I don't want to do that kind of work; there's no value-added for me, so I'd like to buy those kinds of components. But I can't even buy things that help me print easily in Smalltalk, let alone worry about financial calendars."

Even were the components available, it is not certain that the building-block approach offers more than just a base structure for business applications. What is needed is a clean, practical way to allow persons who are not OO specialists to modify components, yet at the same time constrain the ways in which those components may be changed. Cameron provided an example from Wall Street: "In our world, financial instruments are changing every day. You need to be able to give a trader the ability to change the terms and conditions of a **financial instrument** to suit a client's needs. But if the trader at the next desk is trading U.S. government bonds, the last thing in the world you want to give him is a piece of software that allows him to change the terms and conditions of the **financial instrument**! So we have to solve both problems."

The attraction of software components is that they offer a way of leveraging OOP efforts beyond the scope of a single application. Unfortunately, it is not clear to what extent reusability should be considered a normal or desirable offshoot of industrial applications. Stevens crystallized the issue: "What should be the ecology of software? What should be the recyclability ratio of code, for example? If you take an old code and you chop it up into its component parts and lay them on the virtual ground, what should grow from those virtual software components?" Unless the components were designed with considerable vision and skill, they may constrain rather than facilitate future applications.

The use of software components also raises serious issues with regard to intellectual property. The term "software ICs" has been applied to draw the analogy between reusable objects and hardware components (see [3] for a detailed description). However, the transitory nature of software causes some fundamental problems in the areas of ownership and valuation. As Brad Cox pointed out, "Silicon ICs are made out of atoms, and that makes them hard to copy, so you can build a very robust enterprise like Intel by selling hardware ICs. Software ICs are made of bits and can be copied and transported at the speed of light. That breaks down the fundamental economic engine ... We need to solve the problem of ownership and revenue collection for goods made out of bits instead of atoms."

Ownership will obviously be a key concern if software components are to

become a widespread vehicle for development. As objects are acquired, subclassed, and re-released in new forms into the software community, totally new structures will be needed for dealing with the problems of relative and derived ownership. Annick Fron referred to the experience of a group of European contractors involved in developing a graphical interface for signal processing design. One contractor used Smalltalk GUI components, then found that "it took two years for the lawyers to decide which parts were really property of the contractor."

Existing valuation strategies, too, are based on the concept that software is sold in integral, identifiable units. This will no longer be true when software is made up of components that may have been created by vastly different organizations that might not even know of each other's existence. Cox described an experimental approach, devised by the Japanese, where it would no longer be the mere acquisition of software that would trigger the flow of revenue. Instead, cost would be incurred just when that software is executed. This would make copy protection irrelevant, but would clearly require global mechanisms for revenue collection and distribution. The concept of objects would have to be extended to include some consciousness of how often and by whom they are activated. Moreover, it's not certain how such a structure could be made understandable to end users.

Similar issues are already plaguing the field of electronic publications. For almost two decades, proponents have offered a variety of suggestions on how fees could be assigned and collected on the basis of the information actually accessed. No coherent, widely acceptable solution has emerged, however, and it does not appear likely the issues can be resolved simply by encouraging service providers to implement their own mechanisms. As Stevens put it: "The difficulty is that we're trying to invent a new kind of currency, a new kind of transaction. The existing marketplace is not the appropriate vehicle for experimenting with new types of transactions."

Several of the panelists disputed the software IC analogy and were skeptical of the contribution to be made by component libraries. For example, Nugent maintained that he didn't "believe in reusable software components, at the component level ... The reason that hardware ICs were reusable was because they were documented well. You could pick up a book off the shelf ... and find out exactly the performance characteristics, the physical requirements, and every single operational characteristic it had. All its attributes and all its behavior were completely documented and they were frozen because it was hard atoms, not bits. That's what enabled hardware reuse--not necessarily the ability to make the components, but the ability to document them well."

Other issues remain unresolved, including responsibility for the correctness of software components. Verification and validation procedures are difficult enough in an environment where one group is responsible for developing an application. Software constructed of off-the-shelf or modified components simply will not fit our current structures for valuation and liability.

Nor are software components the only form of reuse. John Vlissides commented: "The funny thing about reuse is that it's sort of like AI--if we're doing it, it's not reuse. And you know, we reuse code all the time. Any time you make a system call you're reusing code. In fact, any time you run a program you're reusing code. But that's sort of taken for granted and people moan and groan about not getting any kind of reuse anymore. There are different levels of reuse. I think code reuse is the lowest and least interesting form, and the highest, most important level is design reuse. That's something we need to focus on more."

Another key may be to consider the scope of reuse. While generalized components are necessary if a wide variety of organizations are going to reuse code, requirements are not as stringent when reuse involves just the original development group. The benefits of reuse can be more accessible in

this situation. According to Ungar, "Reuse within one group of people pays off just in the development time of one program, because it never works the way you want it to the first time. When you want to change it to work right, it's much easier to change if each thing is only in one place. So I really think there's a faster payoff than we might expect."

Why OT Hasn't Been Widely Adopted In Industry

Given the close match between the advantages offered by OO and current trends in business management, it's disappointing that OT has not gained more significant support among industrial software developers. Some of the problems can be blamed on misconceptions due to the relative newness of the technology (see "Is Object Technology Still 'Emerging'?"). Nonetheless, there are some significant obstacles that will have to be overcome before OT becomes a standard for industry applications.

The most immediate difficulty is the fact that software support for OOP is not uniform or widespread. There is still a lot of argument about whether so-called "hybrid languages," those that add OOP constructs to a procedural foundation (such as C++, Objective C, Object Pascal, or Ada), are acceptable substitutes for true OOPLs. Generally, the panelists thought that hybrids encourage bad habits (see the discussion of learning mechanisms). By obscuring the object structure, such languages can also prolong the development process. Ungar described a company "that had to do a project and floundered for two years in C++, then finally some grad student did it in Smalltalk in six months. The anecdotes are out there."

Unfortunately, the more purely OO languages are limited in terms of both the number of companies that offer compilers and environments, and the number of hardware platforms on which they are sup ported. This creates a paradox, as Ungar explained: "The new languages that give you the best OO features are supported by a handful of companies, at most, which makes it difficult for people who are betting their enterprise on [the result]; a lot of people simply have to use more widely supported languages. But the widely-supported ones are incremental (i.e., hybrid); their very nature robs the **object-oriented** paradigm of a lot of its power."

Moreover, existing languages are neither consistent nor interoperable. That means the choice of language may have long-term implications about how an application will be able to interface, if at all, to other applications. It also generates considerable confusion among OO programmers, who find that although Smalltalk, C++, Objective C, and CORBA, for example, all have "objects," the nature of objects varies significantly among them.

The same difficulty plagues users of higher level object systems, such as OO databases (OODB). The current situation is precarious, Dave Maier said, because "people don't want to commit to object databases where every vendor is the single source of the product." Moving data to a new system is time-consuming, even when good tools are available. If the source of an OODB product then dries up, the lack of standardization across products means that the effort would have to be repeated.

Support is entirely lacking for several characteristics that are growing in importance. Current OO systems don't encapsulate any information on object reliability, performance, or resource utilization. As Stevens commented: "For scientific users to incorporate a set of objects developed by another group, they have to have confidence that the code is efficient and that it uses resources in a predictable fashion. Current OT frameworks just don't provide mechanisms for doing that." He predicts the lack of support for real-time constraints may cause even more problems: "I have a concern that systems will be built in the next few years to address National Information Infrastructure applications that have serious real-time requirements--like the emergency response for telemedicine, or even multimedia distribution mechanisms--will look at **object-oriented** technology today, say it can't handle real time requirements, and therefore will not use it, will

base their development on more traditional real-time oriented technologies. Ten years from now, the legacy systems that we'll have will be very difficult to deal with. We talk now about large scale financial or business information systems being legacy systems, but those are trivial compared to having a million-node distribution for real-time multimedia with petabytes of image data and trying to convert that to some new technology. The **object-oriented** research community needs to move as quickly as they can to develop a framework for making **object-oriented** technology able to support performance-sensitive applications."

Security capabilities also are lacking. Although a number of groups have begun to discuss the issues of how to generate revenues from OO systems, there has been no serious attempt to add privacy attributes to objects. Yet privacy may be a key issue in future, large-scale applications. As Cox pointed out: "Computerized patient record systems generally get presented in terms of how wonderful it is going to be for doctors to be able to look at everybody's data and perhaps even report back to the insurance company or other company. What about things that patients might not want their company to know, like the results of their AIDS test?" Although the concept is neither new nor unique to OT, the capabilities that OT offers for building extremely large and complex systems will bring to the fore issues of data privacy and integrity. Moreover, the encapsulation mechanism which is at the foundation of object systems may be ideally suited for implementing some of the constraints that will be needed to enforce security and privacy, as well as performance standards.

Finally, it is surprisingly difficult to measure object systems. Reliable measurement units for predicting progress, assessing productivity, and evaluating cost are not currently available for OO designs and applications (see [11] for a discussion of the issues). Fron compared the situation to other development methodologies: "When you have a large project in procedural code, you know that each programmer can write eight lines per hour, for instance, so you have a coarse-grained idea of how much time you need. But models for costing simply don't exist for object programs." Vlissides expanded on the idea: "People have tried to come up with software metrics, and they always degenerate to lines of code, which is exactly the wrong measure for OOP. You want to reduce the lines of code, you don't want to give people incentives for writing more!"

These are serious technological problems, and must be addressed by the OT community over the next few years. They ultimately may prove insignificant, however, in comparison with the human factors issues that deter industry from adopting object technology.

Why Training Is Such a Critical Issue

Most investment dollars spent in converting to OT--and the key to the rate of payoff on that investment--can be attributed to education and training. It is not just a matter of learning a new programming language (that's the easy part), nor adopting new software development techniques and gaining familiarity with tools that support them. Although there has been some debate on just how comprehensive the learning process must be (see [5] for a summary and [14] for examples), the panelists concurred that OT represents a radical paradigm shift. As Ungar commented, "by the time you're done with it, it changes your whole way of thinking." Getting to that point can be painfully slow.

The concept of decomposing problems into roles and developing objects that encapsulate those roles is profoundly different from previous emphasis on processes as sequences of actions. Cameron described the problem: "We've trained people for 25 years that every problem is really a procedure. The world works according to procedures; we've brainwashed our users that way as well as our systems people. The difficulty is that some people quite sincerely believe they are doing it a new way, but in fact they're not, because when confronted with a new problem, they automatically go to the

mode of thinking that they're comfortable with." Just bringing people to the awareness they are relapsing into known patterns is a critical first step in shifting paradigms.

Related problems arise when people attempt to learn OT through a hybrid language. It is difficult, particularly for a novice, to switch mental modes when most of the program is constructed along familiar, procedural lines. Vlissides commented: "You know, I'm a C++ programmer, and I've always been. I've never written a Smalltalk program in my life. But I think C++ is exactly the wrong language to start people off with. Basically, if your manager hears about **object-oriented** programming and decides that this is the way the world is going, what he'll do is he'll give you Stroustrup [12] and a compiler and say, 'Here, be wonderful.' And you, the hapless programmer, realize you can't be wonderful, but you have to be productive. You have to show something, so what you do is slip back into the way you were programming before." Since languages such as Smalltalk [7] or Self [13] make it hard to program in the classic procedural style, they encourage the programmer to develop an **object-oriented** style.

An even better approach might be to defer learning any OOPL until the basic principles of objects are understood at a higher level. Maier pointed out, "I'd like to see the first course have no programming language per se in it. I'd like to present a suite of objects and say, 'Let's think about how you put them together to solve problems.'" Other panelists concurred, noting that with this approach, beginners are still assembling primitives to form object systems, but not at the same level they would in a typical OOP course. Instead of relying on the primitives supplied by a programming language, students would use objects as the basic primitive from which larger systems are constructed. In focusing on the building of systems rather than programs, novices would be exposed to the fundamental properties of objects instead of implementation details.

Ultimately, the real learning hurdle will be the issue of good design. It's comparatively easy to learn how one defines an object and encapsulates it in a software packet containing the data and methods associated with it. It's also easy to learn how to hook multiple objects together. Basic methodologies for decomposing a problem into objects are also mechanisms that can be taught and learned in a relatively straightforward way. However, it is far from clear how to teach people the concept of good design. Yet the success of any object system ultimately depends on design decisions such as what responsibilities an object will have or how objects will encapsulate business procedures and policies.
Vlissides offered an analogy to clarify the problem: "If you look at these design methodologies, none of them are going to make Picassos or Frank Lloyd Wrights out of people. They're generally good for capturing good designs. They can be used as a way of modeling your system to analyze what the objects should be, and you come up with an initial design that is a simulation of reality. But if you look at mature **object-oriented** designs--ones that make good on the promise of reusability, flexibility, extensibility--they have lots of objects in there that don't reflect anything in the original domain model. There are objects that get introduced late in the design process to enhance flexibility and to reduce the interdependencies between objects so that you don't have this 'object spaghetti.' They reify abstract concepts that you didn't think of initially when you were modeling your system. So the methodologies are good for capturing good designs, but they presuppose that you have a good design to capture."

It's possible to outline some of the characteristics of people likely to respond to OT. For example, researchers from IBM studied the effects of previous experience on the ability to learn the principles of OO design and their implementation using Smalltalk [10]. They used the terms "dogmatism" (defined as "markedly entrenched in a single approach to software development") and "openness" to characterize the learners' previous

exposure to design techniques and different styles of programming languages. Not surprisingly, the study found that "dogmatic students do poorly, while inquisitive students and those with broad programming backgrounds do better" [10]. Even the broadest exposure to design and implementation strategies, however, is not enough to guarantee success in learning how to apply OT to an industrial application domain. Another pair of researchers studied how professional programmers, with experience developing industrial OO applications, approached the design of an object system in an unfamiliar domain. They concluded the absence of domain knowledge "reduces the task to manipulations in a formal, abstract system" and results in poor design choices because the programmers have trouble differentiating between important and unimportant attributes [4].

One problem is the lack of good didactic examples based on real-world needs. Cameron complained that "I'm getting tired of ATM examples and the vending machines. It is something that I can read, and I say, 'well, yes, I see how they did that.' But when I go back to do something in a manufacturing setting--or if I'm going to automate an oil refinery--it doesn' t mean I'm going to automatically make the transition." Fron described the efforts of the European Smalltalk Users organization to host experienced OO programmers, discussing how they developed industrial applications. She commented on the success of this program, stating that "basically the object paradigm is programming by example, so it's no wonder that [the best] training is training by example."

The difficulty in both teaching and learning OT is that the quality of an OO design is difficult to evaluate. Design methods have reached enough maturity that virtually anyone can learn how to decompose a problem into roles and interactions. But problems are subtle, stemming from the fact that the selected roles and interactions are inappropriate, or that they are incompletely or incorrectly defined. Consequently, the model is flawed and a great deal of effort can be invested implementing and using that application before the flaws are exposed.

Panelists agreed the ideal way to learn OT is through some form of apprenticeship. The real issue, as Fleming expressed it, "is teaching people good judgment. I don't know if anybody else here has learned how to fly an airplane, but there's this thing called aeronautical decision making. You can't learn that from a book. You have to go up in a plane and fly around and have your instructor sitting there as you do something really stupid. You feel it in the seat of your pants, or you get scolded, but you find out by experience ... I don't believe you can learn judgment from a book, I don't believe you can learn **object-oriented** design from a book. Ultimately I don't think there's a substitute for a coach."

Unfortunately, the strategy may not be easy to implement, again for reasons of human factors. A strong coach has knowledge not only of the material to be learned (in this case, OT), but also of the domain in which that material will be applied (i.e., a particular business setting). It is not easy to find such persons. Moreover, as Cameron cautioned, the availability of a coach does not necessarily mean that he or she will have the requisite interpersonal skills. "It may not be someone's motivation to help other people to become as expert as they are in objects. So while those certainly are ideal qualifications, and I would agree that I find it much more pleasant to learn, and perhaps more productive to learn by having someone who's a raving expert sitting at my elbow, coaching me through it every day, I think it's unrealistic to assume that this technology is going to be able to become very broadly deployed within Industry if that's the only model we have for training people in objects."

It is not always necessary to have an experienced coach, nor is it certain that the perspective of software engineering is important in embracing OT. Cline described an example that was just the other way around. His group of non-computer sciences was involved in developing some distributed

applications and decided that OT would be a reasonable mechanism. They began by identifying the fundamental roles, then progressed to how objects should interface with one another. Cline described his surprise at the result: "By the time they had actually implemented something, even the first rough code, they'd actually done some design, some analysis, some redesign, and some implementation ... I stood back and said, 'Do you realize what you are doing?' They said, 'We're just implementing this piece of code.' I said, 'You're doing software engineering. If I had told you at the beginning of this project to go use software engineering to do this, would have been thrown out of your office.' But just by the fundamental assumption to use **object-oriented** methods, the rest of it was required. What's interesting is that we started out with domain experts and we now have people who do software engineering."

Regardless of the strategy employed, it is clear that human factors issues create the most significant barrier to the adoption of OT. Although the quality and availability of instructional materials, design methodologies, software tools, and even domain-specific coaches certainly contribute to the process, they only touch the surface of the problem. What is needed is fundamental education in how one approaches design, not just training in design mechanisms. Cameron explained the issue: "Unless we want to wait 20 years for this to be broadly used, we have to recognize that we have a retraining issue which, in some way is an untraining issue as well. We have an existing population who knows how to deliver information systems. Their experience tells them, 'I know how to do this stuff.' Now we're asking them to do this stuff in a way that is fundamentally different from what they know to be successful." The dilemma is that, in the new paradigm, it is necessary to gain a relatively high level of expertise before any success can be realized.

Moreover, the education process cannot be confined to just software developers. To make good on the promise of OT, it must permeate a vertical segment of the company structure; only then will the object model truly reflect enterprise objectives and priorities. Like any radical change in business management, the transformation must affect developers, users, and policy makers. The points of difference between OT and traditional methodologies are too dramatic to survive a slow infiltration from the software group up. As Cameron concluded, "We have to retrain experienced people at every level--not just the ones who are building the applications, but the ones who are managing them, the ones who are measuring them, giving them their incentives, and evaluating them. Unless those people make this shift, we're going to have to wait until they all retire, which I don't think we can afford to do."

Ungar summarized the issues that make training such a problem for businesses that wish to move into OT. In doing so, he drew another analogy: "Looking at the next 10 years, I see object technology as a big jump from the old style of procedural programming. It's as if you're walking along a path and you come to a stream. If you try really hard, you can jump across the stream. But if you play it safe and dance from rock to rock to get across, there's a pretty good chance you're going to slip and fall off and get wet. And I see a lot of the incremental [hybrid] languages as diluting a lot of the benefits and producing a lot more problems than either a clean pure procedural style or the new **object-oriented** style. There's a large mental leap that has to be made. I'm worried that in the next 10 years, a lot of people are going to try crossing the stream by going from stone to stone gradually and are going to have disasters; for example, when they try using objects but don't have garbage collection, they're going to have lots of pointers from one object to another but no way of making sure that when they free an object there aren't pointers still referencing them. I see this as a serious dilemma, with no easy answers. It's hard to psych yourself up to just jump across that stream, and it's a hard transition if to OT. But OT could be dead and buried in 10 years if most people try the stone to stone approach-because we'll have a bunch of angry, wet, dissatisfied users."

Critical Issues for OT Research

In looking ahead, the panelists identified several factors that could keep OT from gaining widespread acceptance in industry. Because they are critical to the future success of OT, these areas should be priorities for active research and development. Some remain open research questions and will necessitate high levels of investment. Nevertheless, it is unlikely that any significant proportion of industrial users will be able to apply OT successfully to enterprise and software modeling until these issues have been resolved.

Scalability of object implementations: Scalability is frequently cited as a major obstacle in the design and construction of very large object systems. Since the term has been applied in many ways, panelists were asked to discuss the implications of scalability in their individual domains of expertise. What emerged was a clear concern for how well existing mechanisms will extend to the massive numbers of objects anticipated in future systems. That concern stems from several considerations related to the computing resource infrastructure.

Resource management becomes increasingly difficult as OT applications grow to span multiple computing platforms. CORBA was cited as an important first step in dealing with the management of large-scale, distributed object systems. The CORBA specification itself, however, does not address the implementation issues that ultimately will determine scalability. Anand Tripathi drew a comparison with architectures for network file servers: "When we talk about the scalability of a network file server, the question we ask is, can a given file server architecture go from supporting, say, 10 client workstations to 100 to 1,000, and does its performance remain within certain specified bounds? Now, CORBA--or I should say, distributed object management--is going to be a very big thing in the next 10 years. CORBA is an architecture, a specification for conformity and interoperability; but the scalability issues will be in the implementation of the protocols. For example, when I refer to a distributed object, I don't care where it is located. Whether it is in Seattle or New York, somehow the system will find it. So the question arises whether the protocol for the object request will be scalable when dealing with anywhere from ten objects to a million objects."

Another issue that remains unclear is how many objects individual servers will be able to handle, and how those objects should be managed. Cline described the problem faced by a large industry wishing to exploit OT on a national scale, choosing as an example the power industry's desire to monitor power consumption and balance power production accordingly. This would involve an object for every basic consumption unit (every power meter), a scale that could be on the order of two million for a single power company. The model would be more useful if patterns of consumption could be analyzed at the level of individual subunits (e.g., appliances in a house). It simply wouldn't make sense to manage all them in a single global layer, since clearly the sub-units of one home can't affect or interact with those of another. Cline compared the situation to the early history of the Internet: "It got started very much the way the WorldWide Web is getting done today--if you needed a connection and you knew somebody that had one, you made a deal with them to get a connection and we got this tremendous spaghetti of networks. When you tried to get from one place to another, at a certain size it just fell apart. We fixed that by putting things in places called routers, similar to CORBA's 'brokers.' But one of the things about the Internet that allows it to scale is that they have a concept of geographic locality. If you're in the same domain or the same state, you're in the same service provider, and that allows the system to scale. In the object world, it doesn't make sense to have something based on geographic locality, but there has to be some form of locality on which you can build the architecture or it won't scale."

The performance, or actual resource utilization, of individual OO

applications is of equal concern. Current object systems are at a significantly smaller scale than the ones proposed by the panelists. Stevens pointed out the designer of an application involving perhaps a million cooperating objects will need to know with some assurance that the system will actually be able to function. "In order to do that, we have to look at the basic models for objects and incorporate more information that will enable us to build or compose performance models of the application as we're constructing it. That has important significance, certainly, in scientific applications. But in business applications, too, you want to know as soon as possible if the path you're on is not going to have acceptable performance." The ability to include performance characteristics as part of object attributes is needed to address this part of the problem.

Another facet is the fact that software optimization is increasingly important as object systems grow in size and complexity. This creates something of a paradox. Dennis Gannon described it: "Workstations and microprocessors are getting very fast, so we're seeing a lot of concurrency at the instruction set level. For **object-oriented** languages to exploit this type of architecture, the compilers have to do a lot of whole-program analysis. That means analyzing the entire class library to see how it's used in the particular application. So we've got a problem where we can't optimize without doing a lot of work on the compiler, and the compilers are going to run forever, and so why bother to optimize?" The distribution of objects across computing systems exacerbates the problem. New approaches to optimization must be devised so that it is possible to achieve reasonable levels of performance with access to what may be only a small portion of the code. This will require mechanisms to abstract the performance characteristics of object subsystems and make them available to compilers that are using the subsystem as part of a larger system.

Interoperability: Even at today's relatively small scale of object applications, there is a growing need to connect dissimilar object systems so that components can cooperate to solve a problem. The CORBA specification, developed by the OMG group, makes a start in that direction by defining a general architecture for how intersystem communications can be carried out. The most recent version, CORBA 2.0, focuses on the issues when two systems housed on different vendor and operating system platforms must interoperate, and reflects the initial experiences of protocol implementors. However, CORBA is only a first step, and does not yet address a number of the more problematical elements of interoperability.

As an example, Gannon described an application he is involved in developing, where radio astronomy data drives a large-scale simulation, which in turn drives a virtual reality engine. CORBA makes a good vehicle for linking the three systems at the level of rather coarse service interactions, but he explained that "When I've got a thousand threads of control that are each encapsulated in objects, and they're coming and going with very short lifetimes, I need a finer-grained architecture than CORBA provides. I also need some nice interface between the finer-grained model and the CORBA model. I'm concerned this is something we're not going to have developed."

The notion of interoperable, dissimilar systems raises the spectre of property and privacy rights as well. There has been little progress in this arena, in spite of the fact that it is absolutely critical to the widespread expansion of object services. As Cline put it, "If we don't come to grips with the privacy issues, the property issues, the mechanisms for payment and so forth, then 10 years from now we're still going to have a lot of wonderful demonstrations but we won't have any real applications."

In general, OT technology needs to be brought up to speed quickly if it is to gain any real credibility. The underlying problems are not new. Tripathi summarized them: "The real world is concurrent. The real world is distributed. In the real world security is an issue. In the real world persistence of data is an issue. And the real world consists of different

vendor platforms." Until object systems can handle this kind of environment in a reasonably seamless fashion, they will not be considered production-level technology.

Tool and language support: As described earlier, the fact that language support is not consistent across multiple platforms is a serious deterrent to the general adoption of OT. Until now, the let-a-million-flowers-bloom philosophy has prevailed, with the result that no language base has emerged the clear winner. Stevens suggested the most likely source of a standard will be the "Microsoft analogy." He suggests that "if someone wants to make their particular model for object technology a standard, they should just print 50 million CDs at a cost of maybe $20 million, and they can buy the entire market by saturating it with their development kits." Appealing to the mass market will require some serious revising of current development systems.

The panelists agreed that commercial products are still a long way from the level of simplicity necessary to appeal to large numbers of users. Ungar seized on the car as an analogy: "You know, it's not an easy thing to operate an internal combustion engine and get all those sparks to fire at the right time, and get the right amount of gasoline and the right amount of air to go in. But all you have to do is turn the key, press the gas pedal, and it all works. What's important to you--namely, how fast you're going and where you're going and maybe when you want to stop--is exposed, and what's not important to you is hidden. Now, today's object systems mostly do a terrible job of that. What's important to you is, 'what can I ask this object, and then what can I find out from what it answers?' Not anything about how big it is, where it is, when I have to throw it away, or what header file I need to use it." The hiding and exposing of information should not be entirely static, though. Certain users, or certain occasions, call for the ability to change objects. What's really ideal is to provide a simple system that can, when appropriate, be opened up to allow modification. Such systems do not exist in the current marketplace.

Another area which promises to yield substantial benefits to the OT user is visual programming. In general, visual representations are powerful because they offer capabilities for abstraction and hierarchical hiding or exposing of detail that simply cannot be accomplished within the linear constraints of text. A significant amount of current research in the visual programming community is focused at support for object systems (see [1] for a survey).

Current commercial environments for OOP include graphical elements, but do not really support visual representations of the basic object concept. Ungar described the problem: "I love Smalltalk, and it's made a huge contribution to the field. But if you look at the screen of Smalltalk, you won't see any objects! You'll see windows and tools and browsers and clever ways of making the most of the real estate on the screen. But you won't see any darned objects. Interfaces could be done differently, to actually fool you into thinking of the objects as real. Even simple things would help, like having the same object only appear on the screen in one place at a time." He also advocated more sophisticated mechanisms, like moving objects on the screen in animated form "to make them seem real and physical and alive. We need a system that's responsive, so when you try something, it acts right away. After all, there are human thresholds on the perception of cause and effect. If something takes more than a few seconds, you won't really associate it as cause and effect unless you think really hard about it. So we have to have systems that respect human thresholds about these things, and are responsive enough."

Visual representations are susceptible to problems with scalability. As Vlissides commented, "If I have a system with 10,000 objects interacting in different ways, I'm not going to get a big benefit out of seeing all ten thousand of those objects strewn all over my screen. The tangibility thing is good to a certain level, but beyond that you need to start leveraging the abstraction powers that a computer system can give you to go beyond the

merely physical and to go towards a more abstract level. And that's the only way that you can take advantage of the capabilities of the computer.' Visual abstraction and focusing--using such mechanisms as zooming, panning, and representations that vary in level of detail--are known in the research world, but they have not yet appeared in commercial products, nor have they been tested against the dramatic scaling problems of very large object systems.

Finally, there is a serious need for more tool integration. Currently, even a comprehensive programming environment forces the programmer to learn and use distinct tools to carry out design, analysis, implementation, documentation, execution, and other tasks. Moreover, in most cases the tools do not interact to any significant extent. Cameron maintained this is totally inadequate for meeting the needs of OT: "We cannot have our artifacts of analysis sitting over here, and our artifacts of implementation sitting over there. We need to recognize the reality that we're going to be learning things all the time, whether we're doing analysis, design, or implementation, and that what we have learned has to be reflected back into each of those aspects. That implies for me that there is no distinction between the application and where the design intelligence is, where the documentation of that design intelligence is, and where the documentation of the analysis decisions is."

Scalability of OO design techniques: In the discussion of performance scalability, Vlissides pointed out that scalability actually included two problem areas: "One is dealing with the saturation of resource requirements: how do you push the limits of the resources that you have? The other is dealing with the ever-increasing numbers of quantities of responsibilities that you're taking on as your system scales, and how you deal with that issue."

OOD techniques have not yet evolved to the point of addressing scalability. Consequently, the human factors issues described earlier can grow to unmanageable proportions as object systems expand. Cameron commented that "When I think about scalability, I don't think so much in terms of the things like volume or distribution problems or other kinds of technical problems, but about scalability in terms of the development of systems--the sizes of the projects that we can support, our ability to manage the process of systems development where the end result is going to be a heck of a lot of code, any way you look at it." For many businesses, the primary attraction of OT is that it offers an alternative to the large monolithic applications that form today's legacy problem. It is not yet clear, however, what kinds of techniques and tools will allow people to understand, debug, and reuse very large object systems.

For OODBs and distributed systems, it is the behavioral nature of systems where objects persist over long periods of time that causes problems. Managing the evolution of such systems is difficult because in addition to legacy code, there is now legacy data. As Maier explained, "When you have large amounts of persistent data, possibly not even all on-line, and it's been created under an object regimen, how do you hope to accommodate changes in the code over the lifetime of that system?" The problem increases as persistent objects accumulate, and is in addition to the problems caused by resource limitations.

There is a very real possibility that at some scale, object systems will simply surpass the bounds for applying effective OO design techniques. As Vlissides pointed out, "Everybody here agrees that in order to do a viable **object-oriented** design there has to be iteration. The larger your system gets, the less luxury you have to actually re-implement significant parts of it. If you're going up to the level of a space shuttle control system, you're not going to re-implement it 15 times."

Support for learning OT: Most panelists agreed that the current style of teaching OT put too much emphasis on language and implementation. Nugent

summarized the problem: "If languages continue to drive this shift [to OT], then we are really barking up the wrong tree. The promise of OT is not in what language we choose, but in how we attack the problem."

As described earlier, the group was clearly concerned about how novices can be introduced to the ideology of design quality. One suggestion is that more effort be made to find large-scale examples; in particular, examples that go beyond the step of illustrating basic implementation techniques. As Gannon commented, "Our educational systems have to change so that we think about the design process in the large by examining very large examples. Frank Lloyd Wright didn't learn to be a great architect by studying plasterboard. He learned by looking at architecture and thinking about architecture in the large. From the university side, we have to really rethink education if we are going to succeed in all this."

Revisiting the concept of apprenticeship as a way of learning by example, the group analyzed what the contribution of the coach was and how that might be accomplished through some other mechanism. Cameron reflected: "The benefit of a coach sitting next to you is not so much that the coach can tell you what the right way is to do something, but rather watch you while you do it wrong and then tell you that it was wrong, how it was wrong, and how it could be right. When we do any kind of training in a corporation on any topic--business or organizational or whatever--the characteristic of an industrial-strength training course is that it provides you with that kind of feedback mechanism." She advocated tutoring software that could fulfill the coaching role without the need for one-on-one personal interaction. "It is curious to me that people who are software developers, and who are continually talking about expanding the role of software and the use of user interfaces, intelligent agents, artificial intelligence, expert systems, and so on, can't come up with some expert systems that can coach people through learning a new paradigm for analyzing their problem spaces."

Expanding on the concept of expert tools, Maier suggested "We need tools that somehow capture and enforce the design rules, saying that you are able to use this component in this way, but if you were to limit your design space and constrain it in this way it would be better. You know, a tool that's sitting there saying, 'Yes, that's an okay way to compose these things; no, that's not an okay way to use those two pieces together.'" Stevens compared that concept with the rule-based tools available for design of hardware ICs: "Engineers who are designing something are free to go outside those rules if they accept that the circuit they're building might have some problem. But it forces a conscious decision when one's using the technology outside of spec. And it's that conscious decision that's more important than whether or not you're constraining the person. A person can always violate it, but in doing so, they've made a conscious effort to understand what the tradeoff is." Clearly, existing tools are a long way from providing expert services of this nature. As the panelists pointed out, however, tool support that would facilitate the learning process could be a very sweet incentive to managers faced with the difficult and time-consuming task of retraining development personnel.

Conclusions

As a group, the panelists clearly believe that OT offers real promise for moving business in a positive direction. It pushes managers and developers to re-think the current organizational structure and how that structure responds to change. It serves as a nucleus for evolving decision and software structures that are not only more flexible, but also more closely intertwined. Given the environment of constant change that characterizes today's marketplace, this type of agility could be a remarkable boon.

However, the promise of OT comes with a high price tag. The key to realizing a big payoff, according to the panelists, is leverage. As Nugent commented, "A lot of how you view OT depends on the scope of your development. If you are worried about just some vertical applications

living on a deskto[redacted] you have one view of OT. But [redacted] you're worried about distribution, if--[redacted] in my case--you're worried [redacted] bout a $20 billion enterprise that's moving completely to OT for all of its integration systems, you have a slightly different scope on the benefits and the reality of doing OT based development. A lot of companies start and fail in OT because they don't have the right leverage. It's true that when you start to begin development, if you're looking at sort of slice development, you find that you've got to build a whole lot of infrastructure to support OT well. You're building one application on top of this enormous foundation, and there's no real leveraging of that development effort. In our case, because we're looking at a broad spectrum kind of approach, we can put many different applications on top of this huge infrastructure. So the leverage for us is far greater. I think the mistake companies make is that they tiptoe into this."

The affects of OT on human resources is profound, and should not be underestimated. If it is to succeed, a move to OT must be treated as a major paradigm shift, with all the associated problems. That is, a company should not focus just on retraining, but also find ways of modifying the reward structure and overcoming resistance to change. The real problem is not just adopting a new technology, but understanding and managing projects. Therefore, the persons who already have a mature, broad vision of company roles and interactions are those who will find conversion the easiest. As Fleming commented, "I think the problems of understanding the scope of their projects and managing them frankly overwhelm a lot of developers and their bosses, and prevent OT from moving forward and prevent their applications from moving forward."
Finally, it is important to remember that there may be significant discrepancies between OT as a long-term, strategic goal and the tactical objectives that serve as intermediate steps in arriving at that goal. Cline summarized the issue: "Long term, there's the vision of having a new way of doing business, having a new enterprise. But you're not going to do that by walking into your office tomorrow and ripping down the whole institution and starting over from scratch. A lot of what we've got to do over the next 10 years is take the incremental steps that are tactical. But let's not lose sight of what we want to accomplish at the strategic level. In general, the characteristics that we're looking for are large-scale distributed systems of objects that allow us to replace components and are capable of growing to handle new problems. That's the vision that I keep in mind. Yes, I might write a lot of C++ code that I'm going to have to throw away, but that's a tactical decision because I can do it now. It doesn't change the strategic vision."

The vision is a powerful one. How many companies get there, and at what cost, will depend on how well we respond to the technological and human challenges that OT engenders.

IAB Participants
Lorette L. Cameron

J.P. Morgan & Company

Raymond E. Cline, Jr.

Distributed Systems Research and Development

Department, Sandia National Laboratories

Brad Cox

Coalition for Electronic Markets

Read T. Fleming

Cadre Technologies

Annick Fron

AFC Europe
Dennis Gannon

Department of Computer Science,

Indiana University

David Maier

Department of Computer Science and Engineering,

Oregon Graduate Institute of Science and Technology

Alan F. Nugent

Global Process and Information Management

Organization, Xerox Corporation

Cherri M. Pancake

discussed leader
Department of Computer Science.

Oregon State University

Rick L. Stevens

Mathematics and Computer Science Division,

Argonne National Laboratory

Anand Tripathi

Department of Computer Science,

University of Minnesota

David Ungar

Sun Microsystems Laboratories

H. John Vlissides
IBM T.J. Watson Research Center

The IAB Panel was convened at the invitation of Lilia Tsalalikhin of IBM's Object Strategy and Implementation Group and Joseph's DeBlasi Executive Director of the ACM.

Glossary of Object Acronyms

CORBA (Common Object Resource Broker Architecture)

ODMG-developed standard for connecting and integrating object applications running in heterogeneous, distributed environments. Defines the request protocol objects use in communicating across platform and machine boundaries.

ODMG (Object Database Management Group)

Small consortium, loosely affiliated with OMG, established to define a standard data model and language interfaces to **object-oriented**

database management systems.

OMG (Object Management Group)
Consortium of OO software vendors, developers, and users promoting the use of objects for the development of distributing computing systems. WWW home page located at http://www.omg.org.

OO (**Object-Oriented**)

Modifier indicating that the associated noun has features to support role-oriented decomposition, modeling, or construction.

OOA (**Object-Oriented** Analysis) Use of role-oriented decomposition techniques to model a system.

OOBE (**Object-Oriented** Business Engineering)

Application of object concepts to the design or restructuring of business processes or an enterprise architecture.

OOD (**Object-Oriented** Design)

Application of object concepts to the design of software.

OODB (**Object-Oriented** Database)
A database where units of information are defined and managed as objects.

OOP (**Object-Oriented** Programming)

Application of object concepts to the implementation of software, employing either an OOPL or a non-OO language.

OOPL (**Object-Oriented** Programming Language)

Programming language that includes features to support objects, such as data abstraction and encapsulation, subclassing, and inheritance; examples include Smalltalk, Self, and Eiffel. May be a hybrid (incremental) language that extends an otherwise non-OO base language through the addition of OO constructs (e.g., C++, Objective C, Object Pascal, Ada).

PDES (Product Description Exchange Standard)

Small consortium, under the coordination of the the National Institute of Standards an Technology, involved in establishing definitions of standard classes to represent different kinds of industrial products.

IS Object Technology Still "Emerging?"
Pancake: The characteristics of an emerging technology, according to Ken Orr, are that "there is more written about it than known about it, there are more people selling it than using it, and the vendors are making more money from education than from selling the tools"[2]. Even after two decades of experience with object technology, those characteristics seem to apply. IS object technology still emerging, and why is it taking so long to gain acceptance?

Vlissides: Yes, object technology is still emerging. It's not clear how many people who think they're doing **object-oriented** programming actually are doing **object-oriented** programming. I think it's a small percentage, so the signal-to-noise ratio is fairly low.

Tripathi: The noise level has subsided in terms of everyone using different terms; we're now hearing more coherent voices. But when it comes to applying this technology to large-scale projects and systems, there still is no consensus on what methodologies should be adopted.

Cameron: I agree that most so-called OO applications are in fact not OO. That is an issue having to do with how we as analysts approach a business problem. We need to decompose the problem space in a different way in order to use objects effectively. I don't know of a solution for how you make that change quickly--but it has to be made if this technology is going to go anywhere in business.

Vlissides: It just hasn't become a productive medium for the majority of programmers and designers out there. People have a lot of mechanism thrown at them, but they can't put that mechanism to use in systems that make good on the promise of object technology: that is, to gain reusability, flexibility, extensibility, and elegance in their software.

Stevens: I think its already has emerged--the paradigm is alive and well. There's definitely a generational effect in that the younger generation of software developers seem to be very comfortable with making more radical paradigm shifts. And the more innovative development organizations have no problem making this transition.

Cameron: The phenomenon of younger systems developers not having a problem making the paradigm shift, is a function of the fact that they're not really making a shift because they didn't have a paradigm to begin with. So they don't have anything to unlearn. Whereas the hundreds of thousands of developers who are out there earning their living doing systems today in fact do have a paradigm.

Nugent: It really does depend on how you slice the viewpoint: for a certain sort of technology-oriented, user, or the business community at large. The problem is not the technology--it could be any new technology, any paradigm shift. The problem is cultural, it's organizational, it's people-focused. And the difficulty is really understanding how to motivate those people to either embrace the paradigm shift or to get out of the way. Business needs to move on.

Getting a Payoff from the OT Investment

Pancake: Much of the promise of Object technology for large-scale industrial applications centers around the fact that industrial software has been evolving, and the new characteristics match up well with the strengths of object orientation. At the same time, there are new pressures in the costs of converting to object systems.

Consider an organization that currently uses a traditional software engineering development methodology, perhaps the waterfall method. How much is this company really going to have to invest before they realize any payoffs?

Cameron: If you go into an organization and say that the only way you're going to be successful with objects is that you have to invest millions of dollars, retrain all of your people for six months, and do an enterprise-wide object model--and then you're going to get some infrastructure code after two years on which you can then begin to build useful applications--this is not a scenario that will succeed today. You've got to pick a target where there are business benefits to be achieved, and that doesn't mean that it is necessarily a huge project. You start out with something that's reasonably manageable with respect to size, and you use that to sort of get the camel's nose under the tent as far as convincing people that the pain and effort expense of retraining your people and changing the way you do things will have a payoff.

Vlissides: Payoff is the key. As usual, the payoff is proportional to the investment, and there are three levels of payoff/investment in object technology that I've observed.

At the level where you have the lowest investment, you just develop a one-off, get-it-to-work kind of application. That's the entry level. If you

use object technology as opposed to, say, standard procedural methods, you may get some benefit in that you are able to better model your domain, you application. You've got better partitioning, some encapsulation. You may even get a little flexibility. But you don't get a huge payoff.
At the second level, your goal is an application that can evolve to meet your requirements. That requires more of an investment; for the first level, you really didn't need people that were terribly savvy in OT. At this level, you want an application that can live, and that requires a much deeper experience with object technology and a much longer lead time and a much steeper learning curve for people at that level of proficiency.

The highest level of payoff also requires the highest level of investment, and that's being able to take the software that you're developing and make it usable across different applications--factoring it out, abstracting it, and making it retargetable to different applications. That's where you really need not just a thorough understanding of objects and a lot of experience with OOP, but a lot of experience with different applications and building them with OT, and taking that experience and distilling it out and factoring it down and putting it in a place where people can reuse it. That's going to take the biggest investment. That also is going to have the least payoff in the short term. But if you stick it out five years or whatever, you may get a huge payoff, especially if it's a domain where you can really leverage a lot of your customer contributions.

Maier: We've been going out and talking to companies like utilities and appeal manufactures and insurance companies who are often fairly conservative adopters of technology. They don't want you to tell them about it. They want to see somebody who has a success story to report of having used OT to do something like the customer information system. Unfortunately, some of the payoff is down the road; you don't really know whether it was a success until five years from now. Was it maintainable? Or did it freeze up and become brittle just like the old stuff? In some cases, the examples that are convincing to them just don't exist yet.

Cline: We've taken a sort of guerrilla approach you can use in trying to incorporate object technology into the rest of your corporation. Sit down with somebody and say, "For this particular application, what do you expect to get from it? How do you expect to talk to it? Don't worry about how we're going to do it; just tell us how you want to interact with it and what you want from it." Once you've defined that interface--and you have to be clear that they don't step over that interface, which is a different problem--you're free to implementation that application in the way that suits your technology best. So after you define the interface for one application and build the application in an **object-oriented** way, you end up with some application-specific objects and some more general objects. Then when you go on to the next application, you pick up the general objects and move them with you. This way you sort of infiltrate object technology, for those people who aren't comfortable with the concept. But the bottom line is that no matter how you implement it, it's got to solve their problem.

Ungar: I'm thinking of some novice who wants to get into objects and is wondering, "How the heck do I do it?" Early in my graduate student life, something happened to me which got me going. Somebody took me by the elbow and sat me down in front of Smalltalk and we did some programming together. He showed me what it was like to think this way and create this way. So I want to put in a plug for any managers, that the best way to switch an organization to OT is to get someone in who knows how to do it and have them sit next to other people one-on-one and build something together. It's the apprentice model. That kind of intense effort at the beginning really seems to make a huge difference in helping people adopt not only the thought processes, but the folklore, all the stuff that's not written down.

Nugent: Right. In an organization where novices are the norm, you can't do it without the direct involvement of experts. There are a number of

different techniques that one can use, but I guess the sad truth is that there aren't many experts. AT Xerox, we're going to invest the money that we're saving on the mainframes that have been out-sourced for the next decade (getting the legacy people out of the way with the legacy systems) and move ahead with a whole new development organization that is going to be mentored and trained and cajoled and rewarded--rewarded!--for adopting the new technology. Now, that's a lot to bet.

Stevens: How much of your proposed success is going to be dependent upon object orientation, versus just good software engineering practices?

Nugent: In fact, we're banking more on the installation of a rigorous software engineering process, a cultural shift in the minds of the people who are doing the work, and a lot of our OO thinking is going to be embodied in OO systems, because that's the nature of the beast. We're not so foolish as to believe that OT solves every single problem. In a sense, our challenge is really not to worry about an OT-based development organization, it's how to capture the profound knowledge of the business in a way that allows us to create requirements that can in fact create systems to address it.

Cox: Part of the problem you must be encountering is what we call inserting OT into the reengineering process, and part of it must be this far more generic issue of organizational learning and change in general. I'm curious about the balance between the two.

Nugent: I think it's fair to say that we have a tremendous organizational learning problem, but that does not make us unique. However, we have not found difficulty in getting the business process people to understand OT at all. They take to it more than the programmer takes to embracing Smalltalk. You know, we're talking to them in their language. They understand it completely. And in our particular case, the metaphor that works really well is the document. When we look at a lot of the business process--whether it's our own or other people's--it's interesting how the document as an object kind of flows through business process.

Understanding workflow in a company is key to discovering what the current state is, then modeling the future work state as kind of a workflow is critically important. Then you perform some gap analysis to figure out how you're going to get there. You need to create the infrastructure today to build the applications of tomorrow. Because we don't know what our business process is going to be 5 to 10 years down the road, we'd better have an infrastructure that can support any kind of development activity that's going to reflect our business process!

In What Fields Will OT Be Important?

Pancake: What areas does object technology show a lot of promise for in 5 to 10 years? Let's talk about new application areas--industries where they haven't picked up on technology yet, but are likely candidates for seeing a real payoff from object systems.

Fleming: There's a ready market in design automation. That's a very interesting area for object technology, because you have fairly sophisticated users who place very high demands on the system. They expect scalability over time, over space. We're starting to see a lot of engineering projects that are international in scope ... For example, right up the is a large company that dominates the commercial aerospace market. Boeing designed a plane totally with CAD/CAM--built only one physical mockup in the whole project. But they did this with a consortium of hundreds of international partners, literally all over the world...Boeing is fundamentally a conservative company, and they used conventional technology to do that. They had to jump through hoops. It was a very difficult thing to do. If something like PDES has been a little farther along, it would have helped them out a lot.

Ungar: I'll bring up [redacted] domain of putting together th[redacted]s rapidly for end users. There's a very interesting match between object technology and the pictures naive people have in their heads about what's going on in the computer. You know, they see something on the screen, they press a button. Pressing a button to change something on your computer screen is a lot like sending a message to an object. Because it is such a nice match conceptually, there's a great opportunity to take all the interactions people have with computers in their daily work, and give them the chance to customize them to make more productive. Today when you want to customize things, all of a sudden you have to be a guru and know a whole lot about it at once.

Cline: As I see it, the two huge areas over the next ten years are the financial services arena and the entire manufacturing area--not only the design analysis but also physical manufacturing. The big buzzwords in manufacturing these days are "agile manufacturing" and the "virtual company," concepts that map directly into **object-oriented** technologies and implementations. But I would also like to point out, as we look at these opportunities, that the fundamental characteristics that these people are looking for from object is somewhat different from what we deal with today. In the future, as you talk about having an economy based on these entities (whether we call them "objects" or we call them something else) they're going to have to be more proactive. Whether they're intelligent agents or subjective objects, you enable them with some responsibility and they get something done for you. That's a different view than we have currently where objects are reactive; you send it a message and it does something and sends something back.

Tripathi: One area where object technology inevitably is going to make a very major impact is Internet resource discovery tools and systems. What we are seeing right now (with Gopher and the World-Wide Web) are the first generation systems. The people who are managing these systems run into enormous problems. For example, on the service side they're based on the file system's naming scheme. The moment the administrator wants to reorganize things, everything starts falling apart ... we need self-descriptive services in objects, where the object can unfold itself, so users interactively can find out the services they want to obtain.

Gannon: Another area where we're going to see greatly increased interest is in the healthcare area, especially for applications on the National Information Infrastructure. This involves telemedicine, and patient/doctor record information, and is very closely related to the evolution of the object database world.

Vlissides: One area where I think you'll see more application of OOT than we see today is in telecommunications. Motorola has a project to bring cellular telephone service to the whole world through 80 different satellites, and they're looking to implement the whole thing using object technology. Telecommunications and connectivity between people is an area that's just booming so fast that it has to apply OT.

Maier: If you look where object databases are being applied now, it's mostly dedicated storage managers for design artifacts or design entities. But I see a real growth opportunity in the MIS domain, where object databases seem to work very well as integration or interoperability layers. You can use them to help build decision support systems or company-wide repositories or ways to get at legacy data--not necessarily assuming that the object database stories everything, but that it gives you objects that let you access relational databases or files or multimedia or what have you. A lot of nice things that come from the behavioral aspects of being able to put methods there along with the data. You can put objects there that can reformat the data into the form that other applications need; they can fill in the gaps. If there's some value there that's not stored, it can be computed (you don't really see the difference; the operations can take care of it). If you have, say, a legacy file format that only a particular application understands, it gives you a place to factor out the bits of

understanding the application has and put them in a place where everybody can get at them.

Cameron: When people would ask me (with some fear and trepidation because they were using technology) how widespread this object stuff was going to be I used to say, in an attempt to be balanced, "well, you know, there are certainly some kinds of applications where you might not be able to exploit the benefits of object technology. If you have extremely stable applications, etc." But I'm having an increasingly difficult time finding any applications like that.

I don't know of any business or industry that isn't coping with increasing levels of change in their environment and an increasing requirement to respond to change. Even the seemingly mundane, deep back office kinds of business applications can't cope. When one of the major auto manufacturers announces they're building their new payroll system using objects, it's because in their environment--where they are dealing with different union regulations and taxation regulations and the various kinds of pension opportunities and all that--they simply cannot cope using conventional technology. I think our discussion would be a great deal shorter had we taken the opposite tack and asked: Where are we not going to see objects?

References

1. Burnett, M.M., Goldberg A. and Lewis, T.G. Visual **Object-Oriented** Programming: Concepts and Environments. Manning (1994).

2. Cockburn, A. The impact of object-orientation on application development. IBM Syst. J. 32, 3 (1993), 420-444.

3. Cox, B., Andrew, J., Novobilski, J. **Object-Oriented** Programming--An Evolutionary Approach, 2nd ed., Addison-Wesley, Reading, Mass. (1991).

4. Dvorak, J.L. and Moher, T.G. A feasibility study of early class hierarchy construction in **object-oriented** development. In Empirical Studies of Programmers: Fourth Workshop. Ablex, Norwood, N.J. (1991). 23-35.

5. Fichman, R.G. and Kemerer, C.F. **Object-oriented** and conventional analysis and design methodologies: comparison and critique. IEEE Comput. (Oct. 1992), 22-39.

6. Gibbs, S., Tsichritzis, D., Casias, E., Nierstrasz, O. and Pintado, X. Class management for software communities. Commun. ACM 33, 9 (Sept. 1990), 90-103.

7. Goldberg, A. and Robson, D. Smalltalk-80: The Language and Its : Implementation. Addison-Wesley, Reading, Mass. (1983).

8. Henderson-Sellers, B. and Edwards, J.M. The **object-oriented** systems life cycle. Commun. ACM 33, 9. (Sept. 1990), 142-159.

9. Jacobson, I. Business process reengineering with object technology. Object Mag. (May 1994).

10. Liu, C.. Goetze, S., and Glynn, B. What contributes to successful **object-oriented** learning? In Proceedings of OOPSLA '92. ACM SIGPLAN Notices (Oct. 1992), 77-86.

11. Pittman, M. Lessons learned in managing **object-oriented** development. IEEE Soft. (Jan. 1993), 43-53.

12. Stroustrup, B. The C++ Programming Language, 2nd ed. Addison-Wesley, Reading, Mass. (1991).

13. Ungar, D. and Smith, R. Self: the power of simplicity. In Proceedings

of OOPSLA '87. ACM SIGPLAN Notices (Dec. 1987), 227-24

14. Ward, P. How to integrate object orientation with structured analysis and design. IEEE Soft. (Mar. 1989), 74-82.

About the Author

CHERRI M. PANCAKE is a professor of computer science at Oregon State University. Author's Present Address: Dept. of Computer Science, Oregon State University, Corvallis, OR 97331, email: pancake(at)cs.orst.edu.

COMPANY NAMES:
Association for Computing Machinery
GEOGRAPHIC NAMES: US

DESCRIPTORS: Computer industry; Software industry; **Object oriented** programming; Associations; Technological change; Business forecasts
CLASSIFICATION CODES: 8651 (CN=Computer industry); 8302 (CN=Software and computer services); 9540 (CN=Nonprofit institutions); 5240 (CN=Software & systems); 9190 (CN=United States)

4/9/3 (Item 3 from file: 15)
DIALOG(R)File 15:ABI/INFORM(R)

00969903 96-19296
The case for expressive systems
Pawson, Richard; Bravard, Jean-Louis; Cameron, Lorette
Sloan Management Review v36n2 PP: 41-48 Winter 1995 CODEN: SMRVAO
ISSN: 0019-848X JRNL CODE: SMZ
DOC TYPE: Journal article LANGUAGE: English LENGTH: 8 Pages
SPECIAL FEATURE: References
WORD COUNT: 5630

ABSTRACT: A new kind of information system (IS) is emerging that will reduce the time to market, help tailor products and services to customers' needs, and make processes more responsive to unexpected events. Expressive systems allow users to adapt quickly and easily to expectations from standard operating procedure. How expressive systems work is described and ways of modifying the roles and structure of IS departments to implement the new technology are suggested. Expressive systems are designed to support exceptions from standard operating procedures, empowered actions, new product ideas, services tailored to individual customer's needs, ad hoc or even temporary changes in organizational structure - none of which, by definition, can be specified up front.
TEXT: On the derivatives trading floor at J.P. Morgan in New York, there is a new information system called Kapital. The trading environment supported by this system is a demanding one: the traders who use it are at the cutting edge of creativity in the financial markets. In addition to trading a wide variety of instruments, they are continually inventing new instruments by combining parts in new ways, "bundling," or fashioning an entirely new set of custom terms and conditions. Conventional applications development approaches do not easily support the degree of systems flexibility required by such an environment.

Kapital employs some of the most sophisticated technology currently fashionable in the world of information systems today: it is based on a distributed client/server architecture, borrows techniques from the world of artificial intelligence, and is one of the purest implementations of the concept of **object-oriented** software in the business community.
Using Kapital, traders can choose the user interface that suits them, from simple business forms to graphical representations. They can perform

powerful financial analytics using complex mathematical models. Kapital accommodates real-time data feeds giving current market information and performs sophisticated portfolio analysis against a variety of market assumptions.
However, what really distinguishes Kapital from other information systems is not the technology, but the fact that it does not attempt to fulfill a specified set of user requirements -- at least in the conventional sense. Rather, Kapital attempts to model the very "language" of J.P. Morgan's trading business -- not only the vocabulary, but also the grammar and, arguably, the style. Kapital implements that language in software, so that the traders can directly express their ideas for new "exotic" tradable instruments. One objective for the system was that, as fast as traders could conceive a valid opportunity for a new kind of **financial instrument** , he or she should be able to directly interact with the system to create that instrument, simulate its performance in terms of risk and profitability, and if satisfied, price and trade it immediately. To do this required that the software present atomic-level financial components and business rules to the traders, along with the capability to manipulate and extend them in new ways.

Kapital is not a programming language in the conventional sense; it bears no resemblance to the so-called "fourth generation" programming languages on the market today, nor even the graphical programming tools now gaining popularity. Its basic constructs are not simply high-level representations of the computer's resources (although these are provided where useful), but the natural constructs and components of the trader's world: cash flows, interest rate scenarios, risk profiles, and so forth.

While giving users direct manipulation of the system's capabilities, Kapital does not eliminate the need for professional programming. A high-caliber team needs to maintain and continuously enhance the business language and its supporting technologies. Furthermore, traders may call on professional programmers to assist them in implementing a more difficult idea, refining a prototype they have developed, or writing a new component from scratch. However, the response time for such requests is measured in minutes and hours, not weeks and months. In part, this is because Kapital provides high-level business capabilities and low-level technical components in one integrated environment. The developers do not have to translate a requirement from a business domain, as with conventional programming, into a different technical medium.

Kapital is an example of a new kind of information system that is beginning to emerge in business, which we have dubbed "expressive systems." Conventional systems support only the standard business operating procedures for which they were designed.(1) Expressive systems are designed to support exceptions from standard operating procedures, empowered actions, new product ideas, services tailored to individual customer's needs, ad hoc or even temporary changes in organizational structure -- none of which, by definition, can be specified up front. These changes can be implemented in a time frame appropriate to the business need -- in some cases, in seconds or minutes, by the users themselves; in more complex cases, in hours or days, with the help of professional programmers; but never in terms of weeks or months.

Expressive systems not only make it easy to implement these changes, they encourage business managers or empowered workers to explore more possibilities for change and thus improve business performance. Using an expressive system within any of these contexts feels as natural as using an electronic spreadsheet program to develop a financial model and then explore "what-if" scenarios.(2) But expressive systems are not mere simulation tools. They permit the user to execute the action through the same system, whether that result is a new **financial instrument** to trade, a new way of routing documents through an administrative function, or a reallocation of work orders between manufacturing plants.

In this paper, we show, with reference to both examples and new theory, that expressive systems will in time become the dominant paradigm for business computing. Implementing expressive systems, however, will require organizations to address new technologies and redevelop many of their existing systems. Furthermore, the implementation and support of expressive systems will require a substantial modification to the roles and structure of the information systems department.

ROLES FOR EXPRESSIVE SYSTEMS

We have identified three specific roles that expressive systems will play in business:

* Reducing the time to market for introducing new products.

* Facilitating the tailoring of products and services to individual customer's needs.

* Making operational processes more responsive to unforeseen events.

(Interestingly, these three roles correspond to Treacy and Wiersema's three dimensions of market leadership: product leadership, customer intimacy, and operational excellence.(3) This suggests that expressive systems have a role in all organizations that seek to lead their markets.)

REDUCING TIME TO MARKET

In many industries, from automobiles to pharmaceuticals, reducing the time to bring new concepts to market is critical. Computer-aided design systems, arguably an early form of the expressive system concept, have encouraged users to explore more design alternatives and better understand the consequences of their actions. The new Boeing 777 aircraft was completely designed on computers using a package called Catia. For the first time in modern aircraft design, it was not necessary to build an evolving full-scale prototype to find out if the pipework and other systems could be routed through the narrow confines of the airframe. The computer simulation provided that intelligence -- the first 777 was built to fly.

The advance of new prototyping technologies, such as stereo lithography, which can create a plastic three-dimensional form from a computer model in seconds, and robotic assembly means that the users of such systems can express their ideas directly into physical form. However, the greater the information content of products and services, the stronger the potential for expressive systems to reduce the product development time.

In a limited range of cases, expressive systems can potentially eliminate the product development process altogether. Arguably this is the case with J.P. Morgan's Kapital system. The ability to create new financial instruments "on the fly" changes the trading paradigm from looking for opportunities to trade each of a set of preexisting instruments to creating the instrument needed to take advantage of each opportunity that arises.

While the financial trading environment is undoubtedly a rarefied one, it is worth noting that there is a clear pattern of innovations transferring from this environment to "ordinary" business.(4) Telecommunications companies and energy utilities, for example, are starting to deploy systems of similar sophistication in order to introduce new forms of billing and new value-added services, in response to rapidly changing market conditions.

TAILORING PRODUCTS AND SERVICES TO CUSTOMER'S NEEDS

Many organizations recognize that to retain their most valued customers, they must increasingly respond to an individual customer's needs. The difficulty lies in being able to do this with something approaching the economy of standardized services; information systems are clearly the key

to this objective. Banks and other financial services have for years been developing "parameter-driven" information systems that would allow them to vary the parameters of a financial product (the interest rate, term, and any discounts) without the need to write new program code. In reality, however, the degree of customization this approach offers is still small, and the professional systems effort needed makes it uneconomical for application to individual customers. One organization that is seeking to break this limitation is Britain's National Westminster Bank (Nat West).

As part of a series of initiatives to define the future of retail banking, the IT strategy department at Nat West developed sophisticated PC--based software that would permit not only the parameters, but the very operating structure, of a bank account to be tailored. Suppose, for example, that a high net-worth customer preferred to have her checkbook sent, not to her home, but to the nearest branch of the bank -- with an advisory letter sent to her home. This apparently simple requirement would be beyond the scope of most parameter-driven systems and would require expensive manual intervention. Nat West's prototype system permits this scenario simply to be drawn out graphically on the screen, creating the necessary supporting systems automatically.

Several things about Nat West's prototype system stand out:

* The software was designed to be used by an IT-literate bank manager or, more probably, by a systems professional sitting next to the bank manager. But, either way, the new type of account is created in real time, typically in response to a customer request.
* The system is graphical, with icons representing all the business constructs that a bank manager would expect to deal with -- customer, interest rate calculator, statement, checkbook, and so forth.

* The system does not present the user with a series of predefined options. Rather, it feels like a well-designed child's construction kit, with which the user can quickly explore and implement almost any idea.

* The user can see the profitability, or otherwise, of the financial product. (This can even be done on a separate screen, allowing the basic product to be designed in front of the customer.)

* Built-in constraints prevent the user from building illegal or nonsensical financial products.

It may be several years before this scenario pervades the bank's branch network, but Nat West is committed to implementing its system and probably has at least a couple of years' lead on its competitors. The difficult part is not designing the graphical front end, but redesigning the deep structure of the core information systems to allow them to be manipulated this way.
Expressive systems therefore perform two important functions in product or service customization. The first is to permit the customization to take place in "real time" at the point of customer contact, rather than later in a back office. The second is to permit the user to explore and understand the consequences (here, in terms of profitability) of the proposed approach or action. Without this capability, there can be no substance to the concept of empowered actions. This theme carries over into the third function.

RESPONDING TO UNFORESEEN EVENTS

In the context of improving operational performance, expressive systems take on two roles: the first is to optimize the refinement of resource use; the second is to facilitate the handling of unexpected events and potentially chaotic disruption.

Manufacturing, logistics, and other key operational functions have been

subject to intensive study and refinement for many years. Both quality and efficiency have been driven up, waste reduced, and nonproductive costs, such as stock, virtually eliminated. The scope for further refinement is narrowing rapidly. But the refinement has brought a new problem: greater potential for chaotic disruption (we use chaotic in the mathematical sense).

American Airlines, whose operating procedures and information systems are second to none, has recognized this. Its systems operational control is the business unit charged with executing the flight schedule and marshaling the many different resources on which it depends. Any flight may draw its plane, flight crew, and cabin crew from three different incoming flights, while baggage handling, gate space, catering, cleaning, and other ground-based resources must also be coordinated. With this level of dependency, unforeseen events, from unscheduled maintenance to adverse weather (not to mention presidential haircuts!), have the potential for chaos. Currently, the antidote means preserving the structure of the dependencies at all cost -- even if this means delaying whole complexes of flights. No one really knows the true cost of these off-schedule operations because of the difficulty of separating the complex costs from routine operations and estimating the impact on customer loyalty, but they are believed to be enormous.

Reducing this cost cannot, by definition, be achieved in the same manner as previous operational refinement. As part of a long-term program to apply the power of IT to this thorny problem, American Airlines has developed new systems specifically to support its flight dispatchers, the individuals who manage a flight from the ground, including all resources and any route changes. Previously, flight dispatchers had to access information systems through the same kind of transaction-oriented interface as the reservation systems. The new system not only provides more direct manipulation of the system, but also helps the dispatchers to explore the consequences of each unscheduled event and each possible response. One feature of the user interface, for example, is a time line that graphically portrays the prior events on which a particular flight depends, future flights that in some way depend on it, and their current status. Dispatchers can instantly see the consequence of shifting the proposed take-off time, in terms of disruption to the schedule, to staff and internal resources, and, of course, to passengers. No airline currently has an effective model of the financial costs of such disruption, or of the impact on future revenue from customer dissatisfaction, but the American Airlines system is a significant step in that direction. One way of looking at this type of expressive system is that it takes the conventional concepts of operations research (including, for example, PERT networks) and makes them an integral part of real-time operational systems.

A second example is Black & Decker (B&D), whose complex manufacturing operations have been based for twenty years on MRP II, the standard for manufacturing resources planning. An increasing proportion of B&D's sales is coming from large and streamlined retail operations, such as Home Depot, that may place orders on the basis of "deliver within seven days or the order is canceled, and we devote the shelfspace to competitive products." With some retail stores four days away by road, this gives B&D just three days to respond. But the MRP II systems require so much setting up and processing time on mainframe computers that they can be run only every seven days.

Moreover, the MRP II algorithm works in one direction only: it converts a master build schedule into a materials requirements. plan and thence into a capacity plan. If there is an unexpected change to the availability of manufacturing capacity, or to the availability of materials, the MRP II system cannot identify the consequences or advise alternative actions.

Against this backdrop, B&D formed a new team for advanced manufacturing technology with the goal of designing the manufacturing control system of

the future. That system, built with the help of a small but innovative software vendor called Intellection, is now in operation in several of its plants. B&D believes that it now has an operational flexibility that not even the Japanese can match. It allows the company to consider the consequences of any event and dynamically explore alternative ways to meet the same requirements. In the not-too-distant future, the system will be accessible from every machine cell in the plant, enabling individual machine operators to identify and understand the consequence of, say, shutting down the machine for an hour's preventative maintenance and finding alternative ways to get the parts made in the meantime.

Thus a key role for expressive systems in high-performance operations is to reduce the potentially chaotic effect of disruptive events. Henry Mintzberg wrote, "When the planners run around like Chicken Little crying, 'The environment is turbulent! The environment is turbulent!,' what they really mean is that something has happened which was not anticipated by their inflexible systems."(5)

EXPRESSIVE SYSTEMS CONTRASTED WITH OTHER SYSTEMS

There are other kinds of information systems, such as decision support systems, and other new approaches to systems development, such as rapid application development, that are seeking to address some of these same business goals.(6) However, there are also some clear distinctions, and it is these distinctions, we believe, that make the expressive systems approach more effective in meeting those goals.

Decision support systems, or executive information systems, have made it possible for users to express their information requirements directly, and their ease of use has encouraged managers both to analyze past performance in greater depth and to simulate better the possible consequences of proposed actions.(7) However, the functionality of executive information systems is typically limited to obtaining and analyzing information -- they are not a medium through which actions can be executed, in contrast to each of the examples discussed above.

Furthermore, creating a decision support system is usually a matter of grafting new software on to the front end of existing transactional systems. The front end, whether an off-the-shelf-package or specifically designed for its purpose, shields the user from the technical details of accessing the underlying systems, provides powerful data manipulation tools, and typically wraps the whole in a nice graphical interface. Newer generations of such packages permit the user to change data stored in the underlying systems, for example, to change a customer's address, and perhaps to invoke standard procedures, such as "issue a statement" without leaving the graphical environment. But the only actions or functions available to the user are the fixed set of transactions that the underlying system was designed to support.

If expressive systems are to support actions not previously specified, the user needs access not only to predefined transactions, but to the building blocks from which new kinds of transactions can be constructed. Most of today's core transactional and other "mission critical" information systems were not written with this objective in mind and do not support access at this component level. Some very modern, large systems include application programming interfaces (APIs), which provide better access to the underlying functionality, but these are intended for professional programmers who will be constructing substantial new software applications. Creating the component level access required to implement the expressive systems concept typically requires a complete rewrite of existing core systems.

Within the IS community, the biggest competitor to the concept of expressive systems is probably rapid application development (RAD)(8) -- the nay techniques for dramatically reducing the lead time to develop new

systems. Some instances of RAD deploy similar technology to that in expressive systems (including client/server and object orientation). Some use conventional systems technology and programming languages but deploy different management techniques such as intensive workshops involving users and developers (sometimes called joint application development, or JAD), and time-boxed deliverables.
The significant difference between the RAD approach and the expressive systems approach lies not in the technology or the actors, but in the intent of the process. JAD and RAD are fundamentally techniques for getting better agreement and ownership of the required specification, either through intensive user/developer workshops at the start of the process, or through an iterative process of delivering crude prototypes of systems and refining the specification based on user feedback from those prototypes, toward a stable end point.

In the expressive systems approach, the iteration is primarily away from a stable start point. The basic components and capabilities of the system provide the start point, but as individual business units or individual users use them, they will move away from the start point as their needs change.

IMPLEMENTING EXPRESSIVE SYSTEMS

Expressive systems therefore change the concept of an application. Today, the term "application" refers to a collection of programming code and data that together meet a neatly circumscribed and well-defined set of business requirements, such as an order-processing system or a credit management system. Applications account for the greater part of the budget, manpower, and management attention of most IS departments. The applications are supported by a common technical infrastructure, whose costs and management are shared, but these are smaller by proportion. Implementation of an expressive system requires an inversion of this balance, with a much thinner applications layer and a much thicker (or richer) shared infrastructure, which comprises not only technical components but also business constructs.(9)

If they are thin enough, applications can be thought of as a wiring layer. We find that this notion has particular appeal to systems professionals old enough to remember analog computers, in which applications were constructed by wiring together standard components on a plug-and-socket panel. Moreover, some of the PC- and workstation-based software tools most appropriate to building expressive systems (for example, Digitalk Parts, NeXTStep, and IBM's VisualAge) permit software components to be visually wired together on screen.

We could go farther and say that the word "application" changes its sense from a noun to a verb (strictly, the gerund of a verb).(10) Application now refers to the process through which the infrastructure is applied to the needs of a business situation or an individual user, rather than to a piece of software in its own right.
So what does the thicker infrastructure actually contain? The sine qua non for expressive systems is an "uncommitted" software model of the business (or the particular domain of the business that the system is to serve). In microelectronics, an uncommitted array is a silicon chip that is made as a standard component but, in the final stage of manufacture, is committed to a specific customer application, such as the video circuitry for a games machine or the ignition control system of a car. Similarly, an uncommitted software model represents a business in generalized form but can be committed (or recommitted) to a particular product set, market channel structure, or business organization.

For some years, there has been limited implementation of this concept in the form of tailorable software packages, which are now gaining in popularity. Here, the user organization has the ability to choose between a number of options (possibly a very large number), predetermined by the

vendors of the package. In a truly uncommitted software model, however, the designers have not attempted to foresee all the possible configurations. It is like the difference between a model car that can be customized with decals and accessories, and a Lego set.

Designers of children's construction sets face a constant trade-off. Versatility requires more different kinds of components and lower-level components (individual wheels and bricks). Ease of construction demands fewer components, and this typically translates to ready-made a subassemblies, like a vehicle cab or house roof. Designers can partially overcome this by creating powerful high-level components that take on several different roles. This principle is called "abstraction," and it is the key to building powerful uncommitted software models of the business. The following example illustrates why:

Three years ago, the Bradford & Bingley Building Society (roughly equivalent to a savings bank) replaced most of its systems portfolio with a new system, developed from scratch and based on an uncommitted business model. The old system was proving increasingly costly to maintain and needed replacement anyway. The principal objective, and the reason for the choice of approach, was a system that no longer constrained innovation. The chief executive himself stated that he did not want to be told that he could not implement an organizational or product change because the system would take two years to modify -- a clear, if negatively stated, call for an expressive system.

Within Bradford & Bingley's system is a generic products engine, which is built around such a generalized concept of a savings product that it is capable of also serving as a loan or insurance product. Each product is attached, not to a customer, but to a more abstract software construct known as "associate" -- defined as a party with which the firm has a relationship. More specific versions of associate include the conventional notion of customer, but also agents and branches (retail outlets). By making all other parts of the software interface with the abstract or common version of these specific entities, a decision to change an agent-based product to a branch-based product has no external impact. Equally, if the firm decided to launch a specialized savings product for its own employees, which might entail special terms or security arrangements, it merely requires a new specialized subclass of associate called "employee," but no change to the product engine. Bradford & Bingley's system also has generic software engines for document creation, for the management of workflow in an administrative process, and for managing selected groups of associates for marketing.

Abstraction is not a new concept in information systems; indeed, it is a basic principle of data modeling, but there have been few tempts to apply this to the functionality (i.e., the code) of systems. Historically, the view has been that code needs to be written to meet the specific needs of an individual application. To the extent that there has been any reuse of existing code, it has been at a very technical level: reusable subroutines for implementing a complex mathematical function or for managing computer resources. Little attention has been given to identifying generic or abstract business functions and implementing these as reusable components.

The advent of object orientation is changing this by replacing the artificial separation of code and data with the more natural concept of self-contained objects. A software object completely models a component of the business domain: it contains the data that represents that component and all the functionality that may change or interact with that data.

Object orientation has a natural fit with expressive systems in several ways: it underpins most sophisticated graphical user interfaces and facilitates the construction of reusable components. However, the greatest significance of object orientation, and the one least understood by most IS professionals, is its ability to support business abstraction. Two

principles of object orientation apply here. The first is called inheritance, which facilitates the creation and management of specialized subclasses of objects, as in the relationship between associate and customer. The second is called "polymorphism," in which different objects execute different code in response to the same message. A spreadsheet and a word-processed document can both respond to the message "print," but the way they perform that function will be different. Polymorphism simply reflects the reality that, in business, there are fewer things we want to do than ways we want to do them. Expressive systems should provide the support for the generic things we want to do, and the user's application of that capability implements the particular way it is to be done.

THE NEW IS DEPARTMENT

What kind of IS department will be needed to implement and/or support information systems that are based primarily on the expressive systems model? No single organization that we have encountered has yet completed this transition, but those who have partially moved toward expressive systems have had enough experience that we can piece together a plausible model for the future IS department. Organizations that identify with the potential benefits of expressive systems must recognize that implementing the concept is as much about changing the structure and behavior of the IS department as it is about changing the technology.

The first distinction between this future model and the IS department of today is in the realm of values, attitudes, and beliefs. Take the widespread belief that "If only we could get the users to specify exactly what it is they require, then our problems would be solved." The concept of expressive systems, unlike iterative development, is not based on the notion that users have difficulty expressing their exact requirements; it is based on the realization that increasingly users cannot state their requirements completely because they themselves cannot know all the business conditions and events they will be facing.

A second distinction concerns the role of systems themselves. In the future model, the role of systems is primarily to facilitate change. This means that the IS department must anticipate business change. It does not mean that IS must predict business change; rather it means that it must build systems that are resilient to future change through the use of abstraction, componentization, and rewirable infrastructure.

Thirdly, IS professionals must change their current beliefs about the difference between developers and users. The distinction has been historically valid because the user and developer dealt with different views of the same system. The developer's view comprised lines of programming code, data structures, and operating system calls, which together form a high-level representation of the computers resources. The user's view comprised menus of commands, forms to be filled, queries and reports, which together form a representation of the specific business operations being supported. It is the translation between these two representations that makes conventional systems development such a time-consuming, arduous process. Expressive systems resolve the problem by having the developer and the user share the same representation -- a representation of the natural components and structure of the problem domain rather than a specific solution to it.(11)

Consider, for example, a spreadsheet. Its success lies in the fact that its constructs (tables, cells, and formulas) are a very natural representation of the structure of financial modeling problems and are suited to both very simple and very complex applications. If you have used a spreadsheet, you have almost certainly developed an application; moreover, between developing the spreadsheet and using the resulting application, there is no switch in the representation used. There is a clear distinction between the authors of the spreadsheet package (Microsoft or Lotus, say) and the user/developers, but this is not the same relationship as between

conventional systems developers and users.

It is a curious fact that while many IS departments acknowledge the very sophisticated spreadsheet applications within their businesses, few provide any real support for spreadsheet development. Indeed, there is often an underlying cynicism with regard to end-user development in general. Professional developers are fond of quoting surveys demonstrating that 70 percent of all spreadsheets contain some kind of error but are less keen to help reduce those errors.

In the expressive systems era, professional developers still have roles to play, but those roles will change. Some will be deployed in the creation, management, and continuous enhancement of the infrastructure. This, we believe, will divide into three processes: "Business model maintenance" will be concerned exclusively with the uncommitted software model. Its developers will be high-caliber abstract thinkers, and a key issue will be the communication of the knowledge of this model to those applying it. The second process is concerned with the technologies that support the business model. One of the keys will be ensuring that significant new technologies are introduced to the system in the form of generic infrastructure services, rather than as specific applications. The final process we might call technology services, which most closely resembles the IS operations function today, except that it will be concerned with monitoring and improving performance at the level of individual software components rather than just of whole systems.

The professional developers' other role will be to act as mentors within the application process.(12) For example, in the sales and marketing department of Clorox Company, developers have completely adopted this role. In 1985, Clorox installed one of the earliest, truly expressive data retrieval and manipulation packages, Metaphor (which was also an early example of an **object-oriented** system). In Metaphor, users write modules or capsules and then visually wire the capsules together to generate the reports or screens they require. Almost 150 people in the sales and marketing department use Metaphor intensively, and they are supported by between one and three systems professionals, as needs vary. As part of their mentoring role, the professionals look for commonality in capsules written by different users. The professionals rewrite those versions into a standard, robust, more flexible capsule and offer it around to all the users (who themselves often share capsules with each other via e-mail). The notion of improving user-developed systems would be anathema to many systems professionals. But Clorox has found that the net ratio of fUnctionality created to professional systems input exceeds every other application of information systems in the company -- which more or less sums up the case for expressive systems.

REFERENCES

1. E. Dyson, ed., "An Explicit Look at Explicitness," Release 1.0, October 1993, pp.1-19.

2. B. Nardi, A Small Matter of Programming (Cambridge, Massachusetts: MIT Press, 1993).

3. M. Treacy and F. Wiersema, "Customer Intimacy and Other Value Disciplines," Harvard Business Review, January-February 1993, pp. 84-93.

4 T. Malone, J. Yates, and R. Benjamin, "Electronic Markets and Electronic Hierarchies," Communications of the ACM30 (1987): 484-497.

5. H. Mintzberg, Mintzberg on Management (New York: Free Press, 1989), p. 243.

G. S. Haeckel and R. Nolan, "Managing by Wire," Harvard Business Review September-October 1993, pp. 122-132.

7. J. Rockart and D. De Long, Executive Support Systems (Homewood,

Illinois: Dow Jones-Irwin, 1988).

8. J. Martin, Rapid Application Development (New York: Macmillan, 1991).

9. R. Pawson et al., Building the New Information Infrastructure (London: CSC Foundation, 1993).

10. R. Morison, The New I/S Agenda (Cambridge, Massachusetts: CSC Research and Advisory Services, 1994).
11. J. Tibbets, "Object Orientation and Transaction Processing: Where Do They Meet?" (Phoenix, Arizona: OOPSLA 91, Sixth Annual Conference, addendum to the proceedings, 6-11 October 1991), pp. 3-15.

12. B. Nardi and J. Miller, "Twinkling Lights and Nested Loops: Distributed Problem Solving and Spreadsheet Development," International Journal of Man-Machine Studies 34 (1991): 161-184.

Richard Pawson is European director of research for CSC Research and Advisory Services. Jean-Louis Bravard is a managing director at J.P. Morgan & Company. Lorette Cameron is a vice president at J.P. Morgan.

GEOGRAPHIC NAMES: US

DESCRIPTORS: Information systems; Technological change; Implementations; Systems design; Advantages; Organizational change; Corporate planning
CLASSIFICATION CODES: 9190 (CN=United States); 5200 (CN=Communications & information management); 2310 (CN=Planning)

00808099 94-57491
Grasping objects: Get ready for the upheaval
Smith, Carrie R
Wall Street & Technology v11n7 PP: 24-30 Dec 1993 ISSN: 1060-989X
JRNL CODE: WSC
DOC TYPE: Journal article LANGUAGE: English LENGTH: 5 Pages
WORD COUNT: 3143

ABSTRACT: Because of its ability to code programs faster, lower maintenance costs, and reduce the number of bugs that can creep into financial software, **object-oriented** (OO) programming is gaining acceptance among Wall Street traders. It is the only technology capable of keeping up with the increasingly speedy innovation of financial instruments by allowing the programmer to build an object library from which he can pull resuable chunks of code. In addition, OO is revolutionizing the role that traders play in the design of new products and in the tracking of product lifecycles. Nevertheless, OO does have its disadvantages, namely an onerous learning curve, a substantial initial investment, and a modification of the way that programmers traditionally approach projects. In addition, all of these circumstances are occurring in a relatively conservative industry comprised of traders who expect instantaneous results, such as the ability to churn out new products despite serious lags in technological advancements.

TEXT: Roughly 450 Wall Street systems developers packed an auditorium at Metropolitan Life on Oct. 5 to learn the ins and outs of **object oriented** (OO) programming. Organizers had to change locations twice to accommodate the overwhelming response from financial houses either implementing or investigating this technology.

Nearly every major commercial bank, investment bank and brokerage firm sent

developers who support front-office systems, with the largest contingents coming from Citibank, Chase Manhattan and Salomon Brothers. Others came from Smith Barney Shearson, Swiss Bank Corp., the Federal Reserve Bank, Moody's Investor Services, Standard & Poors, Reuters, Quotron and Dow Jones Telerate.

Not only was this the first meeting of OONY--the brand new users' group for **object-oriented** technology in New York--but it was also a chance to hear Adele Goldberg, co-founder of ParcPlace Systems, a spin-off of Xerox Corp.'s Palo Alto Research Center, discuss object-based project management.
This is a technology taking Wall Street by storm.

Benefits are the ability to code programs faster, lower maintenance costs and reduce the number of bugs that creep into financial software. Industry experts claim that OO is the only technology capable of keeping up with the increasingly speedy innovation of financial instruments, especially customized derivatives. "Objects are going to enable the financial community to get their hands around the complexity [of financial instruments] for the first time," says Patti Dock, a consultant and president of Pillar Systems Inc. in Sandy Hook, Conn.

That's because OO eases the process of dealing with rapidly evolving financial instruments by allowing the programmer to build an object library from which he can pull reusable chunks of code. Rather than perpetuate the tradition of coding each and every procedure from scratch, trading firms can encapsulate the properties of **financial instrument** components in discrete objects, which they can use over and over again.

Beyond the technical wizardry, can trading houses make money with OO? Yes, contend the software experts. "Some of the most creative uses of **object oriented** technology are on the trading floor," says Susan Cohen, a senior analyst at Forrester Research in Cambridge, Mass. For one, the ability to rapidly strip in and strip out features to create exotic instruments increases a firm's competitive edge and likewise its profit margin. Secondly, once an object has been created and "shelved" in an object library, its reuse should lead to a reduction of errors or "bugs" in comparison to traditional methods. And fewer bugs should translate into lower maintenance costs.

In addition, OO is revolutionizing the role traders play in the design of new products and in the tracking of product lifestyles. Instead of waiting for programmers to design financial products, some traders can assemble new hedging strategies by simply pointing to objects labeled Interest Rates, Yield Curve, Cash Flows and Currencies. Once the technology has truly taken hold, experts see the chance for the integration between the front-and back-office functions. As functions normally associated with back-office work move onto the workstations, traders will take on a greater understanding of the full trading process.

But for all of the splendor of OO there are several hurdles the financial industry must still face. Beyond the onerous learning curve, which experts claim is scaring off potential users, OO requires a hefty up-front multimillion dollar investment as well as a modification of the way programmers traditionally approach projects. All this takes place in a relatively conservative industry comprised of traders who expect instantaneous results--whereas experts estimate OO users can wait anywhere from one to two years for anything tangible.

Previously, Wall Street churned out new products despite serious lags in technological advancements. Because firms were unable to process the most exotic products, the effects of the disparity could be felt in Wall Street pocketbooks. "One of the most intractable problems for firms is the length of time it takes to develop applications," says Cohen at Forrester. "That gets in the way of firms competing effectively."

To compete in the [redacted]utthroat world of custom der[redacted]tives, dealers must price, hedge and design transactions before software exists in order to book the business. "The standard development techniques cannot keep up with the way the market is evolving," says Jim Rogers, vice president of Sanwa Financial Products (SFP) in New York. SFP, the three-year-old fixed income derivatives trading subsidiary of Sanwa Bank, was in a unique position in the industry when approaching OO. As a newer entity, the firm began to explore OO "from the beginning," says Rogers.

Rogers admits SFP was hesitant to dive head first into OO due to the industry's lack of familiarity and dearth of knowledgeable programmers. But their need for effective technology was stronger. "With derivatives you cannot create a system and walk away because by the time you design and implement it, it is out of date," says Rogers, who chose Montage, object-based software from International Financial Technology (Infinity) Inc. based in Mountain View, Calif.

Today, almost two years after conducting a pilot project and deciding to go ahead with Sun Microsystems's C++ system in their derivatives trading environment, SFP has survived the onerous learning curve and has laid the groundwork for an OO environment. Rogers goes so far as to say OO gave SFP the opportunity to jump into the market faster. "Once you build your infrastructure, it allows you to innovate quicker," he says. "The derivatives market is constantly changing, and that is where the **object oriented** market has tremendous value."

ROLE REVERSALS. Along with a decrease in the turnaround time necessary for systems development, OO is affecting the traditional roles held by programmers. Programmers, who traditionally progressed up the corporate hierarchy and were compensated according to their project functions, are now considered "designers" and "implementers" in the new age of OO. And the differentiation between the two is somewhat sticky for human resources departments.
"The differentiation lines tend to go away," says Dock at Pillar. Dock advises Fortune 50 companies on their strategies, organizational structure, roles and the eventual price tags in making the transition to OO. Dock attributes this change to the nature of the beast--OO demands that all players in a project have a hand in all stages of development rather than working around a multitiered programming structure.

While causing upheaval in programming roles, OO is also revolutionizing the way members of the financial industry traditionally view the makeup of financial instruments. "From the technological perspective, **object oriented** technology changed the way we think about design," says Michael Packer, managing director of Bankers Trust. "It is a set of interrelated components rather than information that flows from box to box."

Bankers Trust is currently utilizing OO in a joint venture with International Business Machines to build the LS2 system, a real-time commercial loan system. OO-based LS2 tracks the early discussion between the borrowers and lenders, the formation of deals, the primary syndication of the deals, as well as the automated funding of loans, servicing, secondary marketing and the trading of the loans, says Rich Freyberg, managing director of loan system development at Bankers Trust. Due to the fact that Bankers Trust is currently reengineering their commercial lending process, "OO allows us to be flexible in the reengineering approach, i.e., we can change our approach," Freyberg says.

OO is also prompting a change in trader functions. While there is still uncertainty as to the ultimate impact OO will have on traders, the revolution in software may spur a change for traders by placing traditional bad-office processes on the workstation. "OO has empowered the traders at the desks to do things that they used to have other people do for them,"

Dock at Pillar.

Similarly, OO is prompting an evolution of the traditional trading environment, says Cedric Packham, vice president and director of information services at ScotiaMcLeod in Toronto. "**Object oriented** technology is allowing us to create some very effective front-office systems," he says. Packham cites the capital markets trading desk Scotia implemented through Decision Software Inc. (DSI) of New York as an example. While Packham declines to attribute the move from bad to front office solely to OO, he acknowledges that "some of the functionality being provided in the front office, such as real time exposure and position management, is moving work from the back office forward." Scotia's interest in OO led the firm to choose DSTS, DSI's real-time, multi-currency trading, position and risk management system. "We were looking for a trading system that could take us through the mid-'90s," says Packham.

While traders begin to reap the benefits of OO and create their own products via their workstations, a prospect that Philip Meese, director of technical services at Mercury Technologies in New York places "in the long-term," many questions emerge. What will happen to product innovation on Wall Street? "I suspect you are going to see a lot more innovative products," says Ron Dembo, CEO of Algorithmics in Toronto. And will traders require increased mathematical competency? Not so, according to Meese. Essentially, "you can map the traders' world out," says Meese. If OO keeps progressing, traders may eventually be able to develop highly structured deals based on elements such as cash flow, date series, yield curves and underlying instruments with which they are already familiar.

But Michael Hanet, vice president of trading systems at Chase Manhattan, is a bit more skeptical about the prospect of traders taking a more active role in the actual programming of new instruments. "Traders won't develop instruments," he says, "unless it is layered in a user-friendly fashion." Chase took on OO three years ago when it purchased Opus by Renaissance Software of Los Altos, Calif. for interest rate derivatives. Though the bank is keeping Opus for the foreseeable future to handle all of Chase's global risk management products, Hanet says the bank is now starting to use VisualWorks, the latest Smalltalk product by ParcPlace Systems Inc. in Sunnyvale, Calif. According to Hanet, VisualWorks will be used for the whole product spectrum at Chase. As an indication of the learning curve involved in OO, Hanet says that though the latest system was introduced to Chase five months ago, the firm is now "barely seeing the changes."

As for the rumors regarding the power of OO to actually erase the more extreme distinctions between front-and back-office functions, Dembo at Algorithmics tends to believe otherwise. "I don't think it is integrating the front and the back office necessarily," Dembo says. "No one has really developed a good back office that is **object oriented**," he points out. Nevertheless, Mercury's Meese claims sights for the future are set on the tasks of front-office and back-office integration.

WEIGHTY DRAWBACKS. For all of its purported miracles, OO does have rather weighty drawbacks. For one, the prevailing opinion in the industry is that taking on OO means taking on a daunting learning curve--one that may have frightened away more than a few potential object users. "Is the learning curve driving firms away?" asks Forrester's Cohen. "Absolutely," she says.

Experts estimate is takes anywhere between one to two solid years for programmers to successfully learn the ins and outs of OO. The same experts estimate the learning curve for fourth-generation languages (4GL) or C is six months. But, according to Dock at Pillar, in that one-to two-year period a programmer learns a lot more about the system than he would about 4GL or C--including design, debugging and testing.

But for all the stress and strain involved in learning OO, experts claim the end results far outweigh any problematic roadblocks. "If the goal of

truly reusable class libraries can be achieved, it is definitely worth the learning curve," says Packham at Scotia. Dock concurs. "If a person truly wants to learn **object oriented** technology, the learning curve is surmountable," she says.

And, apparently, traditional programmers are on equal footing with those entering the market fresh when it comes to learning OO. The deciding factor is the individual's ability to grasp the new approach to programming. "In the past, software was developed in a divide-and-conquer fashion," says Cohen at Forrester. "**Object oriented** technology is a more holistic approach."

Dock also pegs success in learning OO to the individual's attitude toward new technologies. When she conducted programmer training, "success depended upon whether I dragged them in kicking and screaming or whether they came in on their own," Dock says.

Scotia had a double-edged encounter with the OO learning curve, says Packham. In one sense, the firm did not experience the learning curve because DSI brought the OO-based trading system to the firm in production form, which eliminated the need for rigorous programmer training. But, to provide the firm with a better appreciation of the learning curve, two of Scotia's top programmers worked with DSI in the development stages. While one Scotia programmer readily grasped the concept of OO technology, the other programmer experienced difficulty. "This experience showed us that the individuals' background and experience played a significant part in their ability to grasp the concepts," says Packham. "Programmers experienced in user interface more easily adapted to the OO environment."

While the need for technological aggressiveness in learning OO programming is inarguable, Ray Dodd, a principle at Fusion Systems in New York, chooses to downplay the fear factor of the learning curve. "There has been a tendency in the past to present **object oriented** technology as a new mystical technology where you have to forget everything that you have learned," Dodd says. "That is clearly bogus." OO is based on previous platforms, knowledge and ways of doing things, Dodd says.
In addition to the arduous learning curve, a lack of technological OO experience in the financial industry and reportedly immature support tools have also raised concerns. "A lot of people are not ready to go for it," says Hanet at Chase. "It is a major paradigm shift." This reluctance has narrowed the field of experienced programmers--a necessary component for introducing the technology to a firm. The concern is rooted in "the depth or lack thereof in the talent pool," says Packer at Bankers Trust.

OUT WITH THE OLD. Tying expectations for new technologies to the traits of old technology development is causing additional problems for OO's acceptance. First, traders are accustomed to tangible progress during product development. With OO "you spend a lot of lead time developing infrastructure with very little to show," says Meese at Mercury. "Traders are used to having something to show." But, again, experts harken back to the "no pain, no gain" theory. "One of the presumed benefits of **object oriented** technology is that if you do your homework up-front, you get a lot of benefits down the road," says Dock at Pillar.

Firms must be patient and realize that the task of building an object library alone adds to the task. "Until you get the objects on the shelf, you don't have the full benefits of the technology," says SFP's Rogers. "In developing objects you are paying your dues up-front and investing in technology." Meese at Mercury agrees. "There is an investment in infrastructure but once the infrastructure is there the applications are rapidly developed," he says.

Second, Dock prefers to play up OO's role in empowering individuals in the development process, whereas tools were about removing some of the more mundane aspects of human involvement. "It is not about control, and CASE

[computer-aided software engineering] tools were about control," she says. Dock's concern is [redacted] industry's search for tools similar to those for older technologies--tools that may never exist. Regardless, Dembo at Algorithmics says even tools such as compilers for OO still have a long way to go. "The basic tools are still flaky," he says.

Overall, budgeting time and money for conversion to OO is a function of firm size and what applications the firm is seeking to develop, says Hanet at Chase. Financial investment in OO depends on several factors, including whether firms wish to convert all processes to OO or just specific projects and whether firms bring in outside consultants for the transition, says Dock. At the least, large corporations will need to invest millions, "not hundreds of thousands," each year in OO, she adds.

Hypothetically, if a firm is conducting three pilot projects, Dock estimates that the firm could spend in the neighborhood of $180,000 on consultants as well as another $54,000 training internal personnel to be "mentors" on later projects. The other possibility is for a firm to send select internal personnel to an off-site apprenticeship program, which Dock estimates could cost $80,000 per project team. This is all in addition to training for the remaining internal personnel to acclimate them to OO as well as to software costs, which generally depend on a firm's deal with a vendor. "Every firm has very different and personalized goals and objectives in mind when transitioning to objects," says Dock. Regardless, after the first year a firm's OO costs will "absolutely decrease," says Dock, because the first year is spent training people.

So where will OO go from here "OO is still coming of age," says SFP's Rogers. Likewise, Packer at Bankers Trust says there is "a tremendous amount of evolution to come" with regard to OO. But there is the inevitable fact that all technologies eventually gather dust. Despite his declaration that "**object oriented** technology is a pretty big evolutionary step," Dembo at Algorithmics remains steadfastly realistic about the future. "Within five years you will see another paradigm or some major enhancement," he says.

But for now things are looking up for OO. Says Dr. Jeffrey McIver, director of financial engineering at Infinity: "**Object oriented** technology won't be the new kid on the block. It will be the only kid."

AT THE HEAD OF THE CLASS...

Here are the top two **object oriented** (OO) languages wrestling for Wall Street's attention:

C++: Developed by Bjarne Stroustrup at AT&T Bell Laboratories as an add-on to its C language, C++ is virtually synonymous with OO. C++ is referred to as a "hybrid" language, as it is a combination of two programming approaches--the traditional method represented by C and pure OO approaches represented by Smalltalk. C++ is often thought of as the best choice for those familiar with C.

A conceivable drawback to C++ is the financial industry's perception that it is more of a transition language than true OO. "C++ is still a revision of C and not totally **object oriented**," says Jim Rogers, vice president of Sanwa Financial Products (SFP) in New York. SFP uses Sun Microsystems's C++.

Smalltalk: Developed by Alan Kay and a team of researchers at Xerox's Palo Alto research center, Smalltalk is not a spinoff on an earlier language and is therefore considered a more "pure" version of OO technology--pure OO hints of greater extensibility to those in the marketplace. Smalltalk is considered more suitable for previous users of Cobol. Smalltalk's biggest battle will be marketing itself to Wall Street user of C--a healthy portion at the least.

GEOGRAPHIC NAMES: US

DESCRIPTORS: **Object oriented** programming; Financial services; Applications; Advantages; Information technology; Systems development
CLASSIFICATION CODES: 5240 (CN=Software & systems); 8130 (CN=Investment services); 9190 (CN=United States)

4/9/5 (Item 5 from file: 15)
DIALOG(R)File 15:ABI/INFORM(R)

00764834 94-14226
Prometheus unplugged
Young, Jeffrey
Forbes ASAP Supplement PP: 149-150 Sep 13, 1993 CODEN: FORBA5 ISSN: 0015-6914 JRNL CODE: FBR
DOC TYPE: Journal article LANGUAGE: English LENGTH: 2 Pages
WORD COUNT: 1230

ABSTRACT: On the trading floor of the Swiss Bank Corp. in Chicago, 4 floors above the chaos of the Chicago Board of Trade's commodities futures pits, all the action happens on Next, Sun Microsystems, Hewlett Packard, Symbolics, and other workstations - as well as on a growing number of hand held, wireless devices of all kinds. Dwight Koop, Swiss Bank's executive director of information technology, is charged with integrating wireless technology into his company's work culture. Koop says wireless trading will languish until somebody figures out all the pieces and integrates them. Today, Swiss Bank sends its traders onto the floors of the exchanges with print-outs from its internal computer systems. The bank would like to someday equip its traders with handheld machines. If any trading company figures out wireless, it probably will be Swiss.
TEXT: Dwight Koop works for one of the world's largest, and most conservative, Swiss banks. As he attends a meeting in an elegant conference room, the beep from the pair of palm-sized black boxes on the table in front of him is barely audible. Nestled in a leather carrying case and held in place by Velcro fasteners, the Hewlett Packard 100LX, a top-of-the-line programmable calculator-cum-palmtop-computer, is tethered to an Ericsson GE Mobidem wireless modem whose flexible plastic antenna is waving in the breeze from the air conditioning. Both devices are battery-powered. A symbol atop the modem's LCD screen flashes on and off every few seconds, indicating that something is coming in, or going out.

That's not the only thing going on.

In the meeting room, the voice of an executive in Switzerland booms out of the speakerphone. Beyond the glass wall, a mezzanine gallery circles a basketball-court-sized three-level space filled with pods of desks, stacks of computer monitors and the requisite wall of moving ticker symbols, stock prices and news feeds. This is the trading floor of the Swiss Bank Corp. in Chicago, four floors above the chaos of the Chicago Board of Trade's commodities futures pits. Here, in contrast to the frenzy of colored jackets and hand gestures in the pits, all the action happens on Next, Sun Microsystems, Hewlett Packard, Symbolics and other workstations--as well as on a growing number of handheld, wireless devices of all kinds.

This afternoon the group in the meeting room is discussing new software developments that Swiss Bank is planning. But Dwight Koop, 45 years old and executive director of information technology for Swiss Bank Corp., has other things on his mind. A few taps on the tiny keypad move him through a list of 15 E-mail messages that have landed in his wireless mailbox with the latest beep. The HP 100 and modem are small and unobtrusive enough that he can use them without disrupting the meeting. He stops at one message,

pulls it up on the screen and squints at it through his glasses. A faintly worried look passes over his face. A few more taps and a reply shoots out into the ether.

Simultaneously, his SkyTel SkyWord beeper starts vibrating. Koop checks it, then punches his watch, a Dick Tracy-like digital artifact that not only tells the time but also is a paging device. He stands up, closes his equipment, mumbles something about "putting out a fire" under his breath so the telecommuting executive can't quite hear him over the speakerphone, and breaks away from the meeting.

Koop loves the gadgets. But he's also charged with integrating them into Swiss Bank's work culture, and that, he says, won't be easy: "It's not all ready for prime time yet. The service's interface and interaction model is troubling. For someone who has spent years messing with computers, it can be navigated. Far too often I have to fight it [the system] to make it work."

Koops says wireless trading will "languish" until somebody figures out all the pieces and integrates them. Today Swiss Bank sends its traders onto the floors of the exchanges with print-outs from its internal computer systems. The bank would like to someday equip its traders with handheld machines. "It won't be anytime soon," Koop says. "Trading is a contact sport."

Koop's job involves keeping the bank's global trading systems up and running. These are not just any trading systems. Over the past few years, Swiss Bank has acquired the majority of Chicago's O'Connor Partnerships (the final portion of the acquisition is awaiting SEC approval). O'Connor is one of the most highly secretive, technically sophisticated and profitable of the world's trading operations dealing in financial derivatives--products like options and index instruments that are derived from underlying traded securities such as stocks, bonds and currencies. A leader in the application of mathematical algorithms to options, currency and **financial instrument** trading, O'Connor was once known for trying to avoid all publicity. It once shredded the packaging materials for the workstations it bought and required all employees to sign extensive secrecy agreements. Even today the closest a visitor can get to an O'Connor computer screen is the gallery, a good 30 feet from the trading floor.

Creating new bundles of instruments and options to sell in response to customer requests--instantly--is a growth opportunity that savvy traders have been eager to exploit. All of this is much better handled by computers, which O'Connor realized in the mid-1980s when it led the financial community in moving first to Sun machines and later to more sophisticated workstations. But the new trader's edge is literally up in the air.

WALKING AROUND--GLOBALLY

If any trading company figures out wireless, it probably will be Swiss. Swiss Bank likes to push the techno-envelope, for example, buying 500 Next machines a few years ago because it believed in the future of **object-oriented** development systems. "We have no idea if Next will make it in the long run," says Koop's boss, Craig Heimark, Swiss Bank's managing director of technology. "But someone will succeed with objects. And we'll already have a wealth of experience programming in this kind of environment."

Koop enjoys explaining that he has signed 425 nondisclosure agreements with technology companies. "When companies ask if they can qualify us as a beta site," he says, "I explain that they don't seem to understand. We qualify them." This is real money being handled--security is a big issue. "Dial-in modems simply don't exist in our world," he adds quietly.

For all of O'Connor's history of secrecy, Swiss Bank is talking about moving its proprietary risk-management trading systems to customer sites.

The idea is for the company to let customers manipulate versions of its proprietary analysis tools, then sell them the financial products that meet their needs. Enter wireless. "We want our managers to work by walking around--globally," Koop explains. "But we sure don't want these top-level people fumbling for the phone jack and negotiating with the PBX operator to get a dial-out line so they can check the day's market."

For all his complaining, Koop is still certain that wireless will be a key communications component over the next few years. His own love of the toys convinces him.

On another day Koop is sitting in a hotel in Sausalito, at a window overlooking San Francisco Bay and the city skyline. His computer and modem are open in front of him, and the radio signal is strong. The indicator is flashing furiously. His attention is intermittently pulled to the screen. It is more than a little disconcerting to try to talk with him. We may have to learn a new social dynamic in the era of ubiquitous wireless machinery.

Suddenly he chuckles, and pushes the unit over to display a message he's just received. "A bunch of us have an unofficial contest to send a message from the most unusual location," he says. "Have you ever heard of this place?" The message is tough to see in the glare, but when the tiny LCD display is positioned correctly, it reads: "Do I win the contest? I'm sitting at the bar of the Jaguar Club in Atlanta." (The Jaguar Club is one of the more infamous topless bars in the South.) Ah, brave new world. As it turned out, that message didn't win. The winning one was sent from atop Elvis' grave at Graceland.

COMPANY NAMES:
Swiss Bank Corp International (DUNS:48-000-1239)
GEOGRAPHIC NAMES: US

DESCRIPTORS: Swiss banks; Corporate planning; Systems integration; Information technology; Mobile communications networks; Applications; Futures trading; Commodity futures; Case studies
CLASSIFICATION CODES: 9190 (CN=United States); 2310 (CN=Planning); 5250 (CN=Telecommunications systems); 3400 (CN=Investment analysis); 9110 (CN=Company specific); 8100 (CN=Financial services industry)

4/9/6 (Item 1 from file: 9)
DIALOG(R)File 9:Business & Industry(R)

01416703 (THIS IS THE FULLTEXT)
Objects Take Off
(Marcam anticipates delivery of a version of its Protean production and investory software to Pepsi-Cola North American division)
Information Week, n 568, p 14+
February 26, 1996
DOCUMENT TYPE: Journal ISSN: 8750-6874 (United States)
LANGUAGE: English RECORD TYPE: Fulltext
WORD COUNT: 965

ABSTRACT:
Marcam Corp (Newton, MA) is slated to deliver a version of its Protean production and inventory software on March 5, 1996, to PepsiCo Inc's Pepsi-Cola North American division. According to Bruce Richardson, VP for research at Advanced Manufacturing Research (Boston, MA), a technology research concern, Pepsi is one of the several large companies that are seeking flexibility and ease of use by acquiring **object-oriented** enterprise applications. It appears as though 1996 is becoming the year of **object-oriented** applications. Reportedly, for some time, in-house corporate developers, particularly in industries

such as financial services, have used objects, which are described as reusable chunks of software code, to save time in building custom applications. Although this activity continues to increase, the difference between the most recent trend is that companies are purchasing ready-made **object-oriented** applications, which require little or no development expertise. In the event all goes according to plan with a $3 mil project to test Protean at one of Pepsi's bottling facilities, the company plans to install the software to link all of its North American bottling and warehousing activities in a common system for managing output and inventory of the division, which has $6 bil per year in sales. The company has 65 bottling facilities and 240 distribution centers. According to Marcam, the flexibility offered by **object-oriented** software was the key factor in Pepsi deciding to install Protean.

TEXT:
By Doug Bartholomew

IN CHINA, 1996 IS THE YEAR OF THE RAT. BUT in the world of enterprise applications, 1996 is shaping up as the Year of the Object. In what eventually could turn into the largest implementation yet in the nascent market for packaged **object-oriented** applications, Marcam Corp. expects to deliver a version of its Protean production and inventory software on March 5 to PepsiCo Inc.'s Pepsi-Cola North America division. Pepsi is one of several big companies seeking flexibility and ease of use by buying **object-oriented** enterprise applications. "Everybody is suddenly discovering how hot objects are going to be this year," says Bruce Richardson, VP for research at Advanced Manufacturing Research, a technology research firm in Boston.

For some time, in-house corporate developers, especially in industries such as financial services, have used objects -- reusable chunks of software code -- to save time in building custom applications. While that practice continues to grow, what's different about the latest trend is that companies are buying ready-made **object-oriented** applications, which require little or no development expertise. The payoff for companies comes not in developing the original software, but in modifying the application to adapt to a changing business without having to touch the underlying code. The desired result: a nimbler company.

"Adaptability is the key," says Ray Sasso, corporate chief information officer at J.R. Simplot Co., a $2.8 billion food processor in Boise, Idaho.

Information systems managers should expect to see an increasing number of software vendors offering object-based applications. "Objects are the future of the business because they require very little custom programming," says Doug Magill, a consultant and former CIO at Nestle-Stouffer Co. in Solon, Ohio.

In addition to Marcam, a number of vendors, most notably System Software Associates in Chicago and Systems & Computer Technology Corp. in Malvern, Pa., already have customers using their enterprise-level **object-oriented** packaged software. Dun & Bradstreet Software Inc. has an **object-oriented** workflow capability in its enterprise application suite. Other vendors, including QAD Inc. in Carpinteria, Calif., and Sherpa Corp. in San Jose, Calif., will soon release object-based applications.

If all goes well with a $3 million project to test Protean at one of Pepsi's bottling plants, the company intends to install the software to link all its North American bottling and warehousing activities -- 65 bottling plants and 240 distribution centers -- in a common system for managing production and inventory of the division, which has $6 billion a year in sales. The initial project, scheduled to be up and running by June 1, will include some 200 users with Microsoft Windows PCs. That would expand to include thousands of users throughout the company's supply and

distribution network.
The flexibility offe[redacted] by **object-oriented** software w[redacted]
apparently a key factor in Pepsi's decision to install Protean. "Its business changes rapidly, and the company felt that the adaptability and flexibility inherent with our object architecture would make it easier for them to reflect these changes in the system," says Mark Arsenault, an account manager at Marcam in Newton, Mass.

"We see what we are doing with Protean as having a potential for achieving competitive advantage," says Ken Gerhardt, director of application development at Pepsi-Cola North America.

To win the Pepsi contract, Marcam beat out Dun & Bradstreet, which already had a foothold because Pepsi uses D&B's mainframe-based financial software. For Pepsi, installing Protean is a logical extension of the company's substantial in-house **object-oriented** application development effort using the PowerBuilder development package from Sybase Inc.'s Powersoft unit, according to Marcam, which says some of Pepsi's internally developed objects can be integrated with Protean.

Pepsi isn't alone in its desire to use objects to act quickly. Marcam, which has sold Protean since late 1994, recently signed deals with such major companies as 3M Co. and Sun Chemical Corp., as well as with Simplot. With any fast-changing business, no application will last long. "In that context, you can't have an investment in old, inflexible applications that are difficult to maintain," says Steve McClure, director of object tools at International Data Corp., a Framingham, Mass., market research firm.

For example, in financial services, where **object-oriented** development is common, "every day someone's inventing a new **financial instrument**," McClure says. Those companies can't wait two years to respond."

But fast reaction times aren't just for financial firms any more. Simplot, the Idaho food processor, began using Protean at one plant last fall and at a second plant in January. "The object orientation provides us with so much customizability," Sasso says. "It makes it easy to adapt the system to business changes very readily."

Sasso also says it was easy to train workers to use the system to record shipments, update inventory, schedule production, and track orders. Roughly 80% of the workers had never touched a computer before. Simplot figures it will spend $15 million to $25 million to roll out the software to its entire 38-plant operation over the next four years.
Ramsey Sias Co., a Brecksville, Ohio, maker of fruit fillings for companies such as Dannon and Haagen Dazs, began using Systems & Computer Technology's **object-oriented** Adage software in January to manage purchasing and accounts payable. "The company's users can see their business process expressed as objects on the screen, and they are able to grasp the different steps involved pretty quickly," says Magill, a consultant for Ramsey's object software project.

Despite the burst of object activity, observers say technology users have been slow to embrace the software packages because the technology is so young and relatively untested on a scale the size of Pepsi's plans. Others may be hesitant because they perceive no immediate payoff. "It's still at the leap-of-faith stage," says Richardson, the manufacturing technology analyst.

Maybe, but a growing number of big companies are taking that leap.--With additional reporting by Bruce Caldwell

COMPANY NAMES: MAR[redacted] CORP; PEPSICO INC
INDUSTRY NAMES: App[redacted]cations software; Software [redacted]
PRODUCT NAMES: Applications software packages (737263)
CONCEPT TERMS: All company; All market information; Corporate strategy; Orders; Trends
GEOGRAPHIC NAMES: North America (NOAX); United States (USA)

4/9/7 (Item 2 from file: 9)
DIALOG(R)File 9:Business & Industry(R)

01278766 (THIS IS THE FULLTEXT)
Transaction Monitors: THE OPEN VIEW
(Unix transaction monitors help businesses synchronize transactions, implement three-tier client-server architectures, and boost throughout)
Information Week, n 543, p 45+
September 04, 1995
DOCUMENT TYPE: Journal ISSN: 8750-6874 (United States)
LANGUAGE: English RECORD TYPE: Fulltext
WORD COUNT: 1830

ABSTRACT:
Unix transaction monitors synchronize transactions between heterogeneous databases, implement three-tiered client-server architectures, boost the through-put of online transaction processing, and increase the number of users that a system can support. The monitors are becoming more popular. The technology is less expensive than their mainframe transaction-oriented database counterparts, including IBM CICS, and Computer Associates CA-IDMS. CBIS, a subsidiary of Cincinnati Bell Inc, uses both Unix and legacy transaction monitors. It has processed more of its bills and customer service calls with an internally developed system called Precedence 2000. The article discusses Precedence in detail. Sales of transaction monitors will reach $537 mil by 1998, according to the Standish Group International (Dennis, MA), up from 1994's $109 mil in sales. Unix transaction monitors help cut personnel costs as well. More companies need the technology to help them manage three-tiered client-server systems. The number of three-tier systems will increase by almost 75% between now and 1997, accoridng to consulting firm Strategic Focus (Milpitas, CA). Vendors are working to get rid of many of the limitations of the Unix transaction monitors as well. The article goes into significantly more detail.

TEXT:
Unix transaction monitors help businesses synchronize transactions, implement three-tier client-server architectures, and boost throughput. They could even displace mainframe monitors.

By Dan Richman

HOW DOES A bank ensure that a customer's transfer order actually credits the right account while debiting another? How does an engineering firm use a group-ware application to pass off a computer-assisted design for an engine bolt to a supervisor for examination and approval before that design is shipped off to a factory? The answer is the transaction monitor, also known as a transaction manager. Once found only on mainframes, Unix transaction monitors synchronize transactions between heterogeneous databases, implement three-tiered client-server architectures, boost the throughput of online transaction processing, and increase the number of users that a system can support.

Some users and analysts deride Unix transaction monitors as too complex and immature for widespread use. But the technology -- really a form of middleware -- plays an increasingly important role in corporate client-server environments.

Transaction monitors help guarantee that data-processing transactions succeed. Or, if the process fails, they make sure the failure won't damage the integrity of the data. Monitors also allow load balancing, or the spreading of applications among different machines for maximum efficiency. They also offer various remote procedure calls that link the layers in a three-tiered client-server set-up, helping a business choose the right tier for the right piece of the system.

Helping Hands

There are several reasons for the growing popularity of Unix transaction monitors. The technology tends to be less expensive than their mainframe transaction-oriented databased counterparts, such as IBM CICS, IBM IMS, and Computer Associates CA-IDMS. The Unix version also requires fewer administrators, is easier to program, and tends to use cheaper hardware.

"We're getting performance that's just as good as we're used to on the mainframe," says Jim Holtman, VP of system architecture at Cincinnati Bell Information Systems Inc. (CBIS) in Cincinnati. But Holtman and other users have words of caution. They say tools to manage Unix transaction monitors are uncommon or inefficient, can't handle many users, and don't perform batch functions as well as mainframe monitors. They're even worried about the maturity of related functions in the Unix world such as backup and recovery.

That's why CBIS, a subsidiary of Cincinnati Bell Inc., uses both Unix and legacy transaction monitors. Its Unix system is based on the market-leading transaction monitor. Tuxedo, from Novell in Provo, Utah. But the majority of processing at CBIS is still based on the IBM CICS database and transaction monitoring system.

Over the past five years, CBIS has processed more of its bills and customer service calls with an internally developed system called Precedence 2000, built around Tuxedo. With Precedence, CBIS performs billing and online customer services for cellular phone companies. Unlike the mainframe systems at CBIS, Precedence 2000 runs in a distributed fashion, spreading transaction over multiple servers to increase computing efficiency. That's a key difference compared to mainframe monitors. Given the nature of Unix systems, the management of distributed processing is vital.

Precedence 2000 helps 80 service representatives, using PCs running Microsoft Windows, respond to nearly 200,000 customer calls daily. Holtman says the Tuxedo-based system works so well that all new customers are put into it, leaving only long-standing customers in CICS systems. The company prefers the C and C++ development languages, which work well with Tuxedo for extensible, **object-oriented** development. CBIS likes Tuxedo so much that eventually it may try to move all its operations off the mainframe.

But Holtman acknowledges there are drawbacks. "The critics have some good points to make," he says. "In some respects, Unix transaction managers aren't as capable as mainframes. If someone would invent a way to accurately translate Cobol automatically into C, it would sure help us out."

Though Precedence processes 200,000 transactions each day, the mainframe can handle 10 times that volume. CBIS believes the mainframe far outstrips any Unix system at inputting huge volumes of data on tape, performing backup and recovery, and executing major batch jobs and sorts.

Leap Of Faith

More businesses are experimenting -- or like CBIS, running strategic applications -- with Unix transaction monitors. Sales of the monitors will hit $537 million by 1998, a fivefold increase over 1994's sales of $109

million, according to projections from the Standish Group International, a consulting firm in Dennis, Mass. (By contrast, the total market for non-Unix transaction monitors was more than nine times greater last year at nearly $1 billion.)

One user, GE Capital Mortgage Corp., a seller of mortgage insurance in Raleigh, N.C., uses the technology to help link 200 users in 26 branch offices around the country to a Sybase a database in Tennessee. The goal is to record all sales in real time. The company uses Encina, a transaction monitor from Transarc in Pittsburgh, to ensure the integrity of the transactions. The information is stored on a mainframe but soon will be moved to Sybase, and possibly Oracle, under some combination of Unix and OS/2.

"We feel confident enough that we're going to use the mainframe only for storage and take away any transaction processing role," says Stan Patterson, a senior technical analyst at GE CApital. "It's particularly valuable for the heterogeneity it allows."

Ed Wehner, manager of business information services at Memc Inc., a silicon wafer producer in St. Peters, Mo., rebuts the criticism that a Unix transaction monitor can't be used effectively for batch processing. Using CICS/6000, an AIX and HP-UX version of IBM's CICS, Memc runs batch transactions of customer order, scheduling, and labeling processes even while the system is operating online. When Memc used CICS on the mainframe, the company couldn't run batch and online queries at the same time. "There nothing we can't do better under Unix than on the mainframe," Wehner adds.

Managing Growth

Using Unix transaction monitors cuts personnel costs, too. While Wehner used six or seven people to run his mainframe operations, today the same monitoring tasks can be handled by a single analyst.
One of the reasons for the proliferation of Unix transaction monitors over the past five years is that more companies need the technology to help them manage three-tiered systems, clients (containing the interface) connect to applications (containing the program logic), which in turn interact with data-bases and other computing resources.

The number of three-tier systems will grow by nearly 75% between now and 1997, predicts Strategic Focus, a consulting firm in Milpitas, Calif. Three-tiered systems make up only 5% of the total number of client-server applications. By 1997, this architecture will make up nearly 20% of the total, say Strategic Focus analysts.

Essential to the successful implementation of three-tiered architectures is some way of managing the interactions among the layers. Transaction monitors help companies accomplish this, says Ivan Ruzic, Novell's director of marketing for Tuxedo. The alternative, custom-written middleware, is tedious and difficult to create.

One East Coast financial institution implemented a three-tier setup two years ago using Tuxedo. PCs running Windows access applications Pyramid servers, which in turn access data on IBM mainframes. "If we didn't have Tuxedo, there's no way we could have created this system, which is [helping handle] essential functions like cost-based accounting, **financial-instrument** reporting, and commission calculation," says a senior technical staffer at the company.

The major transaction monitors support multiple data-bases. That makes transaction monitors popular not only with companies that operate several database platforms, but with commercial software developers as well. Among the most outspoken is Larry Tanning, president of Tanning Technology Corp. in Denver. "Anyone saying Unix transaction managers aren't ready for prime

time would sound like an idiot to the 4,000 sites where we've installed our software based on t[redacted]," he says.

Tanning's clients agree. Gordon Divitt, president of fund Serv Inc., a mutual-fund network in Toronto, says he's completely satisfied with Transaction Forwarding System, a Canada-wide, three-tier mutual-fund purchasing application that Tanning developed with two partners. "I got what I wanted," says Divitt. "Development was quick. It operates efficiently. It's stable and easy to extend."

But tanning agrees there are still important weaknesses in the technology. Unix system- and network-administration tools lag those for mainframes. Indeed, analysts like Rich Finkelstein, president of Performance Computing in Chicago, say Unix transaction monitors are still far too difficult to install, administer, and use as a base for writing programs. "We need simplification," he says. Roy Schulte, an analyst with Gartner Group Inc., an IT advisory firm in Stamford, Conn., says monitoring tools for Unix transaction monitors aren't up to the level of mainframe products. For some compute-intensive processes, Schulte adds, companies worry that Unix tape handling isn't good enough, and that batch processing is still not up to mainframe speeds.

Future Remedies

Vendors of Unix transaction monitors are working to remedy the limitations. Sometime next year, Tuxedo will integrate more closely with Novell Directory Services, according to Tuxedo marketing director Ruzic. That will let an administrator control Tuxedo-based applications from a single monitor, along with NetWare LANs and devices that run under Novell's Embedded Systems Technology. When Novell starts supporting a Simple Network Management Protocol agent for Tuxedo, administrators will be able to control Tuxedo using HP's Open View, CA's Unicenter, or similar products.
Top End, Tuxedo's rival from AT&T GIS, already can be managed from most major systems-administration tools, says business unit manager Randy Smerik in San Diego. Now, the company will focus on porting the product to Windows NT.

Despite shortfalls, companies are lured by the technology. "There used to be big gaps in functionality [compared with mainframe monitors]," says Mike Prince, director of MIS for Burlington Coat Factory Warehouse in Burlington, N.J. "Today, it's down to the icing on the cake instead of missing an oven to bake the cake."

That oven is getting fancier by the day.

Novell Dominates the Transaction Processing Monitor Market
UniKix Technologies

UniKix	6%
Other	19%
Novell Tuxedo	32%
IBM CICS for Unix	11%
Transarc Encina	15%
AT&T GIS top End	17%

1994 market: $109 million
DATA: THE STANDISH GROUP INTERNATIONAL

Selected Transaction Processing Monitors

Company	Product	Platforms
AT&T GIS San Diego 619-485-2596	Top End	DOS, Windows NT, OS/2, HP-UX, IBM AIX, AT&T GIS SVR4, Sun Solaris, Amdahl/Cray Solaris, Unisys SVR4, Pyramid DC/OSx, Digital OSF1
IBM	CICS	Client and Server: OS/2,

Armonk, N.Y. 800-426-3333		DigitalOSF, HP-UX, [illegible] AIX, Siemens Nixdorf Sin[illegible] Server: OS/400, MVS 390, VSE, Windows NT Client: DOS, Macintosh, VSE, Windows Unix: HP-UX, IBM AIX, Sun with AIX client, Siemens Nixdorf Sinix, others
Novell Provo, Utah 801-429-7000	Tuxedo	Client and Server: UnixWare, AT&T GIS SVR4, Digital OSF1, HP-UX, IBM AIX, Pyramid DC/OSx Sun OS and Solaris, Windows NT, NetWare Client: Macintosh, DOS, Windows, NT, OS/2
Transarc Pittsburgh, Pa. 412-338-4400	Encina	IBM AIX, Sun Solaris, HP-UX, Digital OSF1, Windows 3.1, Windows NT, OS/2
UniKix Technologies Billerica, Mass. 508-663-4170	UniKix	Bull BOSx DG/UX, Encore UMAX V, HP-UX, IBM AIX, Pyramid DC/OSx, Sequent Dynix, Sun OS and Solaris, Tandem IRIS, Unisys SVR4

DATA: INFORMATION MANAGEMENT CO. AND COMPANY REPORTS

SPECIAL FEATURES: Table
INDUSTRY NAMES: Computer
PRODUCT NAMES: Computer display devices, except terminals (357550)
CONCEPT TERMS: All company; All market information; Company forecasts; Corporate strategy; Industry forecasts; Sales
GEOGRAPHIC NAMES: North America (NOAX); United States (USA)

4/9/8 (Item 3 from file: 9)
DIALOG(R)File 9:Business & Industry(R)

01113719
TECHbytes: Big Banks License Cats' Design Software
(Several large banking institutions license Cats Software Inc's CAD financial software)
American Banker, v 160, n 19, p 15
January 30, 1995
DOCUMENT TYPE: Journal ISSN: 0002-7561 (United States)
LANGUAGE: English RECORD TYPE: Abstract

ABSTRACT:
Tokai Bank Europe, Mitsubishi Finance International and Bankers Trust New York Corp have licensed Cats Software Inc's computer-aided-design financial software, Ficad. The software is based on an **object-oriented** approach to programming, and enables traders, risk managers, and product developers to create applications to monitor any **financial instrument**.

COMPANY NAMES: BANKERS TRUST NEW YORK CORP; MITSUBISHI FINANCE INTERNATIONAL PLC (MITSUBISHI BANK LTD); TOKAI BANK EUROPE
INDUSTRY NAMES: Applications software; Banking; Financial services; Software
PRODUCT NAMES: National and state commercial banks (602000); Business software packages NEC (737275)

CONCEPT TERMS: All[redacted]ellectual property; Patent li[redacted]se
GEOGRAPHIC NAMES: J[redacted]an (JPN); North America (NOA[redacted] Pacific Rim (PARX) ; Southern & Eastern Asia (SSAX); United States (USA); Western Europe (WEE); Western Europe (WEEX)

4/9/9 (Item 1 from file: 810)
DIALOG(R)File 810:Business Wire

0444979 BW1051

D & B SOFTWARE: D&B Software Announces Significant New Releases of SmartStream Series

November 15, 1994

Byline: Business Editors/Computer Writers
Dateline: ATLANTA
Time: 05:01 PT
Word Count: 1977

ATLANTA--(BUSINESS WIRE)--Nov. 15, 1994--Dun & Bradstreet Software today announced SmartStream 3.0, a major new release of its SmartStream enterprise suite of integrated workflow-enabled client/server tools and applications which deliver significant functionality and enhancements.

The announcements today of the newly available StreamBuilder, Manufacturing Stream, Distribution Stream, and the major releases of Financial Stream, HR Stream and SmartStream Decision Support, bring to a total of eight new products, support of four new platforms and two languages, introduced this year.

The SmartStream Series is a Microsoft Windows-based client/server-based enterprise solution comprised of integrated business applications, comprehensive decision support tools and a robust workflow-enabled platform that allows organizations to re-engineer their business processes to maximize competitiveness in their industry.

SmartStream 3.0 also includes an improved graphical interface, unparalleled information delivery, new international financial management and reporting capabilities, purchasing and allocations modules, and a business process orientation based on workflow and agent technologies. These improvements address enterprise-wide information management requirements of finance, sales and distribution professionals.

"With SmartStream, D&B Software's integrated client/server applications make a great leap forward to robust, enterprise-wide application functionality," said R. Douglas MacIntyre, president and chief executive officer. "Our client/server applications are designed to address global requirements of the most discerning customers."

SmartStream 3.0 Series includes:

- StreamBuilder -- an application development and customizing tool kit that offers business and systems analysts, programmers and others a proven alternative to traditional development methods.
- Manufacturing Stream 3.0 -- integrated manufacturing applications for discrete and repetitive environments. It supports flexible manufacturing strategies that exploit just-in-time production principles and Kanban inventory management.
- Distribution Stream 3.0 -- integrated distribution applications that support flexible distribution strategies that embrace total supply chain management.
- Financial Stream 3.0 -- a new version of D&B Softwares financial management system that is available in English and now in French. It also includes the availability of purchasing and

allocations modules and major functionality enhancements in asset management, accounts payable, accounts receivable, and international reporting.

- SmartStream Decision Support 3.0 -- D&B Softwares premier analysis and reporting client/server tool includes improvements to the applications SQL engine, as well as enhanced management report development and distribution capabilities. Among the end-user benefits offered in SmartStream Decision Support 3.0 include a major improvement in reporting throughput capacity which delivers a significant performance improvement in the delivery of management reports.

- SmartStream Budget 3.0 -- a new version of D&B Software's integrated budgeting and planning application that includes additional functionality in the areas of creating budgets, viewing and printing reports and integrating with other modules.

- HR Stream 3.0 -- D&B Software's integrated human resources applications has added enhancements in the areas of translation, security and integration with host-based systems.

"Release 3.0 of our SmartStream architecture provides a powerful information reporting and delivery capability that leverages our continued investment in workflow technologies and provides a truly open architecture upon which our customers can build their enterprise-wide business solutions," said MacIntyre.

StreamBuilder

(see separate announcement for more details)

StreamBuilder provides an **object-oriented,** Microsoft Windows-based environment that lets a user create add-ons and customer-specific applications with the same look and feel as SmartStream applications. With StreamBuilder users can also customize their current SmartStream applications and integrate non-D&B Software applications within SmartStream workflows.

Manufacturing Stream and Distribution Stream

(see separate announcement for more details)

Components in this release include: Product Definition; Manufacturing Planning; Inventory Management; Dock-to-Stock/Receiving; Purchasing and Order Management. Product Definition defines and maintains all items and creates and maintains the product structures used by an enterprise. Manufacturing Planning analyzes the current status of each item in inventory, site-by site, and creates replenishment schedules. Purchasing supports the entire procurement cycle from the requisitioner's desk to the placement of purchase orders with a vendor. Dock-to-Stock, Receiving, and Order Management handles the complete management cycle of material movement, from inbound receipt, to customer order management, shipment and invoicing of orders.

Financial Stream 3.0 Enhancements:

Financial Steam 3.0 is an enterprise-wide system providing a core of integrated financial applications that enable cross-functional execution and monitoring of critical financial information and business processes and provide quick feedback for proactive decision making. Financial Stream includes areas traditionally associated with general ledger, accounts payable, accounts receivable and fixed assets. Unlike traditional and competitive financial applications, Financial Stream organizes business objects around activities such as journal processing, payment request, asset management and currency translation. This approach gives customers the ability to easily individualize its business processes.

Purchasing

SmartStream Purchasing enables customers to leverage volume purchasing by allowing them to consolidate purchasing for multiple sites, with multiple delivery dates and multiple ship-to locations, and with multiple accounting distributions to a single purchase user. SmartStream Purchasing includes distributed processing, logical or physical, as well as items, vendors, vendor items, requisitions, purchase orders, blanket agreements and EDI capability.

SmartStream Purchasing was developed to take advantage of workflow and intelligent agent technology, providing increased efficiencies for end users. For example, SmartStream Purchasing can automatically detect a receiving quantity exception and delivers it via mail distribution to the buyer responsible for that particular order so it can be resolved. Following order resolution, the agent sends notification to the receiving department for completion. This eliminates the time spent by users on non-strategic tasks. SmartStream Purchasing is a stand-alone module that is fully integrated with Financial Stream's Financial Records and Payables modules.

Allocations Module

SmartStream Allocations, an allocations module that enables users to more effectively allocate indirect line-of-business or product-line costs and revenues is now available with Financial Stream 3.0. This provides an organization with a better understanding of the true profitability of a unit, division or product category. SmartStream Allocations is fully integrated with Financial Stream's Financial Records module and SmartStream Budget. It also can be used as a stand- alone allocation tool or in conjunction with SmartStream Decision Support.

Matching

Integration between the Purchasing and Payables applications allows a user to electronically match purchase order, receipt and invoice information at the purchase order line schedule level. Selected invoices are compared to the purchase orders and the related receiving documents to establish any exceptions. If exceptions are associated with the invoice, the information necessary to research the exception is viewed on-line. Exception errors can then be corrected by drilling down to the original source system.

SmartStream's workflow facilitates efficient customized exception routing. Since different types of matching exceptions are typically resolved by people in different functional areas, To Do messages with the accompanying matching exception information are routed to the appropriate person based on the type of exception. For example, if an exception occurs when an invoice tries to match against a canceled purchase order, the workflow could be set up to specifically route the error to the user who can investigate and correct the error.

Payables Drafts

Financial Stream 3.0 delivers drafts as an additional payment method available for European customers. Drafts are a **financial instrument** for the vendor that can be discounted or used as collateral for lines of credit. A draft can be created by the vendor issuing the draft, or Bill of Exchange, from his receivables system, or the customer issuing the draft, or Promissory Note, from her payables system.

Combination Validation

Financial Stream 3.0 also includes a "combination checking" function which simplifies the process of adding new account, cost center or product line information. "Combination checking" allows users to easily establish a table of permitted or not allowed combinations of key field information, significantly reducing time spent maintaining the system and posting errors. For example, an oil and gas exploration company, with more than 100,000 wells, has a reporting structure that requires it to add and validate information about multiple locations simultaneously. Since only accounting key combinations needed for posting are created, combination checking can also reduce file sizes.

Asset Depreciation

Financial Stream 3.0 includes enhanced international features to support companies with multi-location Asset Depreciation Range (ADR) requirements. These include the ability to perform ADR accounting and asset retirement reversal and management of changes in scheduled processing, including simplification of the bulk copy procedure and the addition of ADR fields.

SmartStream Budget E[redacted]ancements

SmartStream Budg[redacted] is a fully integrated, client/[redacted]ver budget application which provides an entire enterprise with the ability to shorten the time required to create budgets, automates budget data distribution, enables implementation of a consistent budgeting methodology and provides superior analysis and reporting capability. SmartStream Budget 3.0 has added functionality which includes:

- Integrating with SmartStream Allocations.
- Creating Rolling Budgets, which allows users to continuously budget 12 months into the future.
- Allowing users to prepare budgets more easily through an enhanced workbench windows environment.
- Enabling users to create, view and print reports directly from the workbench.

HR Stream 3.0 Enhancements

HR Stream provides information on fundamental human resources functions from hiring through termination on both a local and global level. These business functions include job and position management, recruitment, employment compensation, training and government compliance. The system integrates with host-based payroll systems and with D&B Software's SmartStream Series of client/server products. HR Stream 3.0 has added functionality which includes:

- Enhanced data security functionality for viewing of sensitive personnel data such as salary information.
- Compliance with international standards which makes it easier to convert HR Stream to other languages.
- Enhanced interface to the host for activities like payroll processing.

SmartStream Decision Support 3.0 Enhancements:

SmartStream Decision Support 3.0 offers new capabilities designed to allow faster information access and easier and broader information distribution.

Improved Data Delivery

Using SmartStream Decision Support 3.0s SmartStream Query & Reporter, customers can create a "distribute by structure" option. This enables customers to distribute common reports to multiple recipients or to deliver reports of varying detail or "views" to multiple sites across the enterprise. For example, a western region sales report listing all sales during the quarter can be sent to the regional sales manager, while streamlined reports listing only particular accounts in that region can be sent to individual sales representatives.

For departments or organizations with requirements to perform analysis and reporting for multiple executives or locations, SmartStream 3.0 now provides the ability to handle multiple management reports in a SmartStream application and simultaneously deliver multiple reports or "information packets" to numerous users simultaneously through any MAPI or VIM-compliant electronic mail system. By enabling users to group reports together by recipient, SmartStream Decision Support is further streamlining the information delivery process.

Pricing and Availability

Financial Stream 3.0, HR Stream 3.0 and SmartStream Decision Support 3.0 is shipping this month. Financial Stream and HR Stream run on Hewlett-Packards HP-UX, Data Generals DG/UX, IBMs RS/6000 AIX, and Sun Microsystem's Solaris. SmartStream Decision Support runs on Hewlett-Packards HP- UX, Data Generals DG/UX, IBMs RS/6000 AIX and OS/2, Sun Microsystem's Solaris, Intel-based Microsoft NT, Digital's Alpha OSF/1 and ICL DRS 6000. Pricing for Financial Stream starts at $250,000. Pricing for SmartStream Decision Support and HR Stream each starts at $100,000. Pricing for StreamBuilder starts at $20,000 per module. Manufacturing Stream and Distribution Stream are priced from $300,000. Pricing depends on system configuration and number of users.

D&B Software, with more than 10,000 customer sites in over 60

countries, is a company of The Dun & Bradstreet Corporation. D&B Software, with more than 2,000 employees, provides a broad range of business software products, tools and services, including decision support tools, financial, human resource, manufacturing and distribution applications.

SmartStream and Financial Stream are registered trademarks and StreamBuilder, SmartStream Budget, SmartStream Decision Support, HR Stream, Manufacturing Stream and Distribution Stream are trademarks of Dun & Bradstreet Software Services, Inc. Throughout this release, other software and hardware products are mentioned by name. In most, if not all cases, these product names are claimed as trademarks by the companies that manufacture the products. It is not our intention to claim these names or trademarks as our own.

CONTACT: D&B Software, Atlanta
Lorretta Gasper, 404/239-3658
or
Copithorne & Bellows, Boston
Tim Hurley/Dave Copithorne 617/252-0606
KEYWORD: GEORGIA
INDUSTRY KEYWORD: COMPUTERS/ELECTRONICS COMED
TELECOMMUNICATIONS

4/9/10 (Item 2 from file: 810)
DIALOG(R)File 810:Business Wire

0444880 BW1687

D & B SOFTWARE: D&B Software Announces Significant New Releases of SmartStream Series

November 14, 1994

Byline: Business Editors/Computer Writers
Dateline: ATLANTA
Time: 14:44 PT
Word Count: 1978

ATLANTA--(BUSINESS WIRE)--Nov. 14, 1994-- Dun & Bradstreet Software today announced SmartStream 3.0, a major new release of its SmartStream enterprise suite of integrated workflow-enabled client/server tools and applications which deliver significant functionality and enhancements.

The announcements today of the newly available StreamBuilder, Manufacturing Stream, Distribution Stream, and the major releases of Financial Stream, HR Stream and SmartStream Decision Support, bring to a total of eight new products, support of four new platforms and two languages, introduced this year.

The SmartStream Series is a Microsoft Windows-based client/server-based enterprise solution comprised of integrated business applications, comprehensive decision support tools and a robust workflow-enabled platform that allows organizations to re-engineer their business processes to maximize competitiveness in their industry.

SmartStream 3.0 also includes an improved graphical interface, unparalleled information delivery, new international financial management and reporting capabilities, purchasing and allocations modules, and a business process orientation based on workflow and agent technologies. These improvements address enterprise-wide information management requirements of finance, sales and distribution professionals.

"With SmartStream, D&B Software's integrated client/server applications make a great leap forward to robust, enterprise-wide

application functionality," said R. Douglas MacIntyre, president and chief executive officer. "Our client/server applications are designed to address global requirements of the most discerning customers."

SmartStream 3.0 Series includes:

- StreamBuilder -- an application development and customizing tool kit that offers business and systems analysts, programmers and others a proven alternative to traditional development methods.
- Manufacturing Stream 3.0 -- integrated manufacturing applications for discrete and repetitive environments. It supports flexible manufacturing strategies that exploit just-in-time production principles and Kanban inventory management.
- Distribution Stream 3.0 -- integrated distribution applications that support flexible distribution strategies that embrace total supply chain management.
- Financial Stream 3.0 -- a new version of D&B Softwares financial management system that is available in English and now in French. It also includes the availability of purchasing and allocations modules and major functionality enhancements in asset management, accounts payable, accounts receivable, and international reporting.
- SmartStream Decision Support 3.0 -- D&B Softwares premier analysis and reporting client/server tool includes improvements to the applications SQL engine, as well as enhanced management report development and distribution capabilities. Among the end-user benefits offered in SmartStream Decision Support 3.0 include a major improvement in reporting throughput capacity which delivers a significant performance improvement in the delivery of management reports.
- SmartStream Budget 3.0 -- a new version of D&B Software's integrated budgeting and planning application that includes additional functionality in the areas of creating budgets, viewing and printing reports and integrating with other modules.
- HR Stream 3.0 -- D&B Software's integrated human resources applications has added enhancements in the areas of translation, security and integration with host-based systems.

"Release 3.0 of our SmartStream architecture provides a powerful information reporting and delivery capability that leverages our continued investment in workflow technologies and provides a truly open architecture upon which our customers can build their enterprise-wide business solutions," said MacIntyre.

StreamBuilder

(see separate announcement for more details)

StreamBuilder provides an **object-oriented**, Microsoft Windows-based environment that lets a user create add-ons and customer-specific applications with the same look and feel as SmartStream applications. With StreamBuilder users can also customize their current SmartStream applications and integrate non-D&B Software applications within SmartStream workflows.

Manufacturing Stream and Distribution Stream

(see separate announcement for more details)

Components in this release include: Product Definition; Manufacturing Planning; Inventory Management; Dock-to-Stock/Receiving; Purchasing and Order Management. Product Definition defines and maintains all items and creates and maintains the product structures used by an enterprise. Manufacturing Planning analyzes the current status of each item in inventory, site-by site, and creates replenishment schedules. Purchasing supports the entire procurement cycle from the requisitioner's desk to the placement of purchase orders with a vendor. Dock-to-Stock, Receiving, and Order Management handles the complete management cycle of material movement, from inbound receipt, to customer order management, shipment and invoicing of orders.

Financial Stream 3.0 Enhancements:

Financial Steam 3.0 is an enterprise-wide system providing a core

of integrated financial applications that enable cross-functional execution and monitoring of critical financial information and business processes and provide quick feedback for proactive decision making. Financial Stream includes areas traditionally associated with general ledger, accounts payable, accounts receivable and fixed assets. Unlike traditional and competitive financial applications, Financial Stream organizes business objects around activities such as journal processing, payment request, asset management and currency translation. This approach gives customers the ability to easily individualize its business processes.

Purchasing

SmartStream Purchasing enables customers to leverage volume purchasing by allowing them to consolidate purchasing for multiple sites, with multiple delivery dates and multiple ship-to locations, and with multiple accounting distributions to a single purchase user. SmartStream Purchasing includes distributed processing, logical or physical, as well as items, vendors, vendor items, requisitions, purchase orders, blanket agreements and EDI capability.

SmartStream Purchasing was developed to take advantage of workflow and intelligent agent technology, providing increased efficiencies for end users. For example, SmartStream Purchasing can automatically detect a receiving quantity exception and delivers it via mail distribution to the buyer responsible for that particular order so it can be resolved. Following order resolution, the agent sends notification to the receiving department for completion. This eliminates the time spent by users on non-strategic tasks. SmartStream Purchasing is a stand-alone module that is fully integrated with Financial Stream's Financial Records and Payables modules.

Allocations Module

SmartStream Allocations, an allocations module that enables users to more effectively allocate indirect line-of-business or product-line costs and revenues is now available with Financial Stream 3.0. This provides an organization with a better understanding of the true profitability of a unit, division or product category. SmartStream Allocations is fully integrated with Financial Stream's Financial Records module and SmartStream Budget. It also can be used as a stand- alone allocation tool or in conjunction with SmartStream Decision Support.

Matching

Integration between the Purchasing and Payables applications allows a user to electronically match purchase order, receipt and invoice information at the purchase order line schedule level. Selected invoices are compared to the purchase orders and the related receiving documents to establish any exceptions. If exceptions are associated with the invoice, the information necessary to research the exception is viewed on-line. Exception errors can then be corrected by drilling down to the original source system.

SmartStream's workflow facilitates efficient customized exception routing. Since different types of matching exceptions are typically resolved by people in different functional areas, To Do messages with the accompanying matching exception information are routed to the appropriate person based on the type of exception. For example, if an exception occurs when an invoice tries to match against a canceled purchase order, the workflow could be set up to specifically route the error to the user who can investigate and correct the error.

Payables Drafts

Financial Stream 3.0 delivers drafts as an additional payment method available for European customers. Drafts are a **financial instrument** for the vendor that can be discounted or used as collateral for lines of credit. A draft can be created by the vendor issuing the draft, or Bill of Exchange, from his receivables system, or the customer issuing the draft, or Promissory Note, from her payables system.

Combination Validation

Financial Stream 3.0 also includes a "combination checking" function which simplifies the process of adding new account, cost center or product line information. "Combination checking" allows users to easily establish a table of permitted or not allowed combinations of key field information, significantly reducing time spent maintaining the system and posting errors. For example, an oil and gas exploration company, with more than 100,000 wells, has a reporting structure that requires it to add and validate information about multiple locations simultaneously. Since only accounting key combinations needed for posting are created, combination checking can also reduce file sizes.

Asset Depreciation

Financial Stream 3.0 includes enhanced international features to support companies with multi-location Asset Depreciation Range (ADR) requirements. These include the ability to perform ADR accounting and asset retirement reversal and management of changes in scheduled processing, including simplification of the bulk copy procedure and the addition of ADR fields.

SmartStream Budget Enhancements

SmartStream Budget is a fully integrated, client/server budget application which provides an entire enterprise with the ability to shorten the time required to create budgets, automates budget data distribution, enables implementation of a consistent budgeting methodology and provides superior analysis and reporting capability. SmartStream Budget 3.0 has added functionality which includes:

- Integrating with SmartStream Allocations.
- Creating Rolling Budgets, which allows users to continuously budget 12 months into the future.
- Allowing users to prepare budgets more easily through an enhanced workbench windows environment.
- Enabling users to create, view and print reports directly from the workbench.

HR Stream 3.0 Enhancements

HR Stream provides information on fundamental human resources functions from hiring through termination on both a local and global level. These business functions include job and position management, recruitment, employment compensation, training and government compliance. The system integrates with host-based payroll systems and with D&B Software's SmartStream Series of client/server products. HR Stream 3.0 has added functionality which includes:

- Enhanced data security functionality for viewing of sensitive personnel data such as salary information.
- Compliance with international standards which makes it easier to convert HR Stream to other languages.
- Enhanced interface to the host for activities like payroll processing.

SmartStream Decision Support 3.0 Enhancements:

SmartStream Decision Support 3.0 offers new capabilities designed to allow faster information access and easier and broader information distribution.

Improved Data Delivery

Using SmartStream Decision Support 3.0s SmartStream Query & Reporter, customers can create a "distribute by structure" option. This enables customers to distribute common reports to multiple recipients or to deliver reports of varying detail or "views" to multiple sites across the enterprise. For example, a western region sales report listing all sales during the quarter can be sent to the regional sales manager, while streamlined reports listing only particular accounts in that region can be sent to individual sales representatives.

For departments or organizations with requirements to perform analysis and reporting for multiple executives or locations, SmartStream 3.0 now provides the ability to handle multiple management reports in a SmartStream application and simultaneously deliver multiple reports or "information packets" to numerous users

simultaneously through any MAPI or VIM-compliant electronic mail system. By enabling users to group reports together by recipient, SmartStream Decision Support is further streamlining the information delivery process.

Pricing and Availability

Financial Stream 3.0, HR Stream 3.0 and SmartStream Decision Support 3.0 is shipping this month. Financial Stream and HR Stream run on Hewlett-Packards HP-UX, Data Generals DG/UX, IBMs RS/6000 AIX, and Sun Microsystem's Solaris. SmartStream Decision Support runs on Hewlett-Packards HP- UX, Data Generals DG/UX, IBMs RS/6000 AIX and OS/2, Sun Microsystem's Solaris, Intel-based Microsoft NT, Digital's Alpha OSF/1 and ICL DRS 6000. Pricing for Financial Stream starts at $250,000. Pricing for SmartStream Decision Support and HR Stream each starts at $100,000. Pricing for StreamBuilder starts at $20,000 per module. Manufacturing Stream and Distribution Stream are priced from $300,000. Pricing depends on system configuration and number of users.

D&B Software, with more than 10,000 customer sites in over 60 countries, is a company of The Dun & Bradstreet Corporation. D&B Software, with more than 2,000 employees, provides a broad range of business software products, tools and services, including decision support tools, financial, human resource, manufacturing and distribution applications.

CONTACT: D&B Software, Atlanta
Lorretta Gasper, 404/239-3658
or
Copithorne & Bellows, Boston
Tim Hurley/Dave Copithorne 617/252-0606

KEYWORD: GEORGIA

INDUSTRY KEYWORD: COMPUTERS/ELECTRONICS COMED
TELECOMMUNICATIONS

02112110 SUPPLIER NUMBER: 19802430 (THIS IS THE FULL TEXT)

Financial CAD's plug and play platform. (Financial CAD Server)(Product Announcement)

Webb, Andy

Wall Street & Technology, v15, n10, p76(4)
Oct, 1997
DOCUMENT TYPE: Product Announcement ISSN: 1060-989X LANGUAGE: English RECORD TYPE: Fulltext; Abstract
WORD COUNT: 2451 LINE COUNT: 00199

ABSTRACT: FinancialCAD introduces its FinancialCAD Server, a single server platform designed to assist dealing rooms with defining instruments and creating integrated financial systems quickly. The product allows users to develop and share models of any type of financial tools, without requiring any programming experience. The platform stems from the Unified & Trade; Data and Object Model, which it uses to manage all transaction and analytical data. FinancialCAD Server also calculates cash flow, risk and valuation data. According to the company, clients will be able to create and implement reliable, scalable applications that will easily integrate

with existing system[redacted] The product costs $15,000 with[redacted] yearly license fee of $5,000 and $100 [redacted] month per user.
TEXT:

Armed with an agressive pricing strategy, Financial CAD is rolling out a single server platform aimed at helping dealing rooms to define new instruments and rapidly build integrated financial systems.

The release of FinancialCAD Corp's (formerly GlasscoPark) new Server must rank as one of the biggest shifts in territory seen to date in the financial IT market. Based in Surrey, British Columbia, the Canadian software company, known for its work with financial derivatives, has previously specialized in producing pricing models, with its current library extending to some 140 items. These were originally produced as Excel add-ins, but have more recently also become available as Visual Basic and C++ run time libraries for those developing inhouse applications. The company has built a strong reputation in this field for both quality and extremely aggressive pricing, but is now up against a very different breed of competitor, albeit in a surprisingly small market.

The new product, called Financial CAD Server, set for release in October, is designed to allow users to instantly create and share models of any type of **financial instrument** without needing any programming knowledge. By using a single server platform, which hinges on the Unified & Trade; Data and Object Model, to manage all analytical and transaction data, the Server will also calculate valuation, risk and cash flow data for those models. The company claims that clients will be able to develop and deploy scaleable, reliable and powerful applications that will easily integrate with existing systems.

But, defining the rather nebulous "financial server" market that FinancialCAD has just joined isn't that easy. While there are plenty of companies that offer toolkits to develop new financial applications, there are relatively few that can offer an automated **object-oriented** environment suitable for the rapid development and deployment of fully integrated financial solutions. Of these firms, Infinity Financial Technology Inc., based in Mountain View, Calif., has (by IT standards) attained almost grandfather status. Apart from FinancialCAD, other players in the field include Integral Development Corp. (founded by Harpal Sandhu on leaving Infinity) and two Nordic companies -- VärdePappersData and Oy Trema.

While the attractions of a structured **object oriented** environment for rapidly developing new financial applications have been the topic of much brave talk in the past, the reality has typically been somewhat in arrears. However, there are at last signs that the technology is fast approaching a feasible and practical maturity. Deals are being done. Integral announced two major contracts this year: JP Morgan completed the development of a front-to-back office emerging markets derivatives application in six months using Integral's Universal Financial Server (UFS). And Bayerische Vereinsbank has taken delivery of UFS as the basis for an exotics system that will integrate to mid and back offices.

Meridien Research, the Boston-based technology research firm, has recently produced a major report on the "financial server" market -("Development of Risk Management and Trading Applications using Object Technology"). Octavio Marenzi, the research director who compiled the report, feels that the timing of the technology's coming of age is certainly propitious. "The (application) development cycle has simply been too long and as new financial instruments have appeared the real bottleneck has been IT. In an attempt to circumvent this, people have developed products using manual processes such as spreadsheets, which is not a very satisfactory approach when it comes to maintaining controls or integrating with other systems."

The financial server approach isn't a universal panacea, though. Some developers, who have made speed of application development the ultimate priority, can create problems for clients further down the line. "While they may get applications out quickly they lack the quality of data structure necessary for running meaningful data queries at a later date," says Marenzi. "With their approach, a new exotic structure isn't stored in the database in a structured format using a data model, but thrown in as a

large binary object[redacted]f you wish to use this data fo[redacted]ther applications (such as risk manag[redacted]nt) it cannot be accessed using[redacted]QL."

Financial server technology is also not necessarily the most appropriate solution for all. "It can be great for larger firms that have the IT resources to get the most out of it," says David Little, president, Derivate Services Corp. "For smaller firms the training aspects can prove too costly and they tend to do better buying off the shelf solutions."

Though the off-the-shelf approach may traditionally be preferred by second and third tier players, there are signs that financial server technology could soon be trickling down to them. The development of proprietary instrument models may not be feasible for them, but the developers of financial server technology have already defined most commonly (and some uncommonly) traded instruments. For the smallest participants, there is now no reason why this technology should not be available to them on a bureau, or even Internet, distributed basis.

Desks can Define Instruments

The fact that this type of server approach allows the trader (rather than a technologist) to create and define financial instruments can also make for more effective data maintenance. Though approaches to this issue vary slightly among the vendors, the general rule is that the creator of a **financial instrument** on the system is also responsible for its maintenance and that of any associated data. As David Glassco, CEO of FinancialCAD points out: "Who better to maintain the FX spot data than the spot FX desk?"

Nevertheless, some see problems in allowing new financial instruments to be defined in the front office. "This approach has the potential to get very messy," says Marenzi. "No matter how well these things are constructed, there's always the possibility that the trader will do something silly and produce some nonsensical **financial instrument** that will have to be disinterred from the database and rectified. It's also very difficult to optimize this kind of approach. If you have a large database with several thousand transactions an hour passing through it, you really need to get into the code in order to do that, which is something that the trader is unlikely to be able to do."

Unified Data Model

The FinancialCAD Server has had a lengthy gestation, starting life as FEO (Financial Engineering Objects) in mid 1991. ("Until we discovered that FEO meant "ugly" in Spanish," says Glassco.) "We quickly learnt that we needed a normalized data model that managed the primary data relationships," says Glassco. "We also realized that we needed normalized analytical interrelationships, so that if (for example) a 90-day LIBOR rate changed it would automatically propagate throughout the entire system across FX, bonds, commodities, equities and related derivatives like swaps and options."

The unified data model in FinancialCAD Server also means that it is extensible. New types of financial instruments can be added and integrated into the model on an ongoing basis. It also provides for unification across an enterprise, so that (for example) the FX team in an institution is using the same curves as the bond team. At the same time, the unified model dramatically reduces the total number of quotes coming into the system by automatically selecting only those quotes that are actually needed to build such things as yield curves.

Apart from the unified data model, the other thing that Glassco cites as a unique feature of FinancialCAD Server is that it has been implemented in a third generation, three tier client/server architecture. "This is something that the financial sector hasn't recognized yet," he says. "Because the Server is based on a distributed **object oriented** framework, companies wishing to scale from a single user system to several thousand and deploy financial objects on multiple machines on multiple sites can do so."

Predictably Glassco is upbeat about the prospects for this technology, but despite the ability to define any **financial instrument** in FinancialCAD Server, he remains pragmatic about the likely speed of take up in the financial industry. "It takes a lot of time and energy for companies to re-engineer their thinking and systems. However, our view of the

financial market is [redacted]at eventually it's going to be [redacted]Visa-like transaction pumping [redacted]twork, with multiple distribute[redacted]components and transactions made across the Internet. We've built Server to play a part in that."

Aggressive Pricing

The pricing of FinancialCAD Server is aggressive to the point of violence, and therein could lie a problem. (A purchase price of $15,000, with an annual license fee of $5000 and $100 per month per user.) There is the risk that major investment banks will refuse to take a product seriously that is priced at this low a level, when they are more accustomed to price tags on this sort of product starting at around $250,000. However, it is clear that the company is aiming at a much broader market. Glassco is unrepentant about this. "We already service some 800 corporate treasuries and want to price the product so that they can all afford it. We surveyed our smallest customers when fixing the price to find what they could afford. Our costs aren't so high that we have to price it at $500,000 and if we do that we immediately restrict ourselves to the top tier banks."

Microsoft, whose treasury department is going to be the first beta site for the Server (and with whom FinancialCAD has close technology links), endorses this pricing ethos. (Though Financial CAD Server will run on any ODBC database, the foundations of the product are pure Microsoft -- DCOM, NT, SQL and Transaction Servers) "You want the exposure and the continual usage of the product," says David Gumpert-Hersh, Microsoft product and technology manager worldwide capital markets, in Redmond. "FinancialCAD wants the Server anywhere there's a machine that can use it -- even in a three-person operation. They are aiming at higher volumes at a lower price with the broadest possible user base."

Gumpert-Hersh is also upbeat about the prospects for the FinancialCAD Server. "Derivatives are getting more and more intertwined with the everyday operations of treasuries. As a result corporate customers are becoming more sophisticated and making greater demands on their banks. Based on that, you're going to have a lot more value-at-risk calculations demanded from directors and board members. The real value of this product is the new architecture, which drives the product to a much higher level. If it can deliver what FinancialCAD already delivers in an add-in form it should be a real winner."

Though FinancialCAD Server has the ability to service the most sophisticated users and their extensive customization requirements, it will also be available in a format that will allow smaller clients to get up and running rapidly. The product will initially ship on pre-built servers -- typically high-end Pentiums with 64MB of RAM. These will have all the FinancialCAD software (and the supporting Microsoft products, which includes SQL, NT and Transaction Servers) fully configured and tested. All clients will have to do is unpack the crate and plug the machine into their network. They will also have the option of having FinancialCAD set the machines and software up to their specific requirements.

Other players in the field have not been inactive either. Infinity launched their FinEx product at the SIA Conference in June. FinEx, when used in conjunction with Infinity's integrated trading and risk management system, afford traders and other users the same opportunity to define new financial instruments in a spreadsheet automatically in a broadly similar fashion to FinancialCAD and Integral.

Infinity's senior product marketing manager Richard Walker is optimistic not only about FinEx but also about the validity of this approach in general. "Platform-based development has been gaining increasingly broader market acceptance. Infinity customers have demonstrated success with the Infinity Platform for some time. Now other participants are hoping to duplicate the platform approach, and have accepted that this technology is more substantial than a flash in the pan. Infinity's approach differs from Integral and FinancialCAD in that our FinEx offering is wholly integrated with the Infinity integrated trading and risk management system, which includes the Infinity Data Model and Fin++ Class Library. "

Like Glassco, Walker sees the technology developing a broader user base. "We've been very successful in selling the Infinity Platform into

organizations with staff capable of developing C++ applications," he says. "However, the majority of end users of those applications have a different skill set -- they're not system developers. What they are good at is using spreadsheet applications to model the financial behavior of the instruments they handle. By having access to the underlying analytics of Fin++ via an Excel interface with which they are comfortable, these end users are empowered to develop valuable financial tools very quickly."

Integral's customers to date have used its Universal Finance Server to replace their existing financial technology infrastructure in building derivatives trading and treasury-wide risk management systems. Unlike Infinity and FinancialCAD, Raj Patel, managing director of Integral Europe, is skeptical about the "one vendor supports all financial products" approach. "It all depends on how you chose to define 'support'. If support includes workflow, security, audit, and full front to back office processing, I don't think anyone can claim to support all financial instruments. Also, the claim that a single data model can support every type of **financial instrument** from a trading and back office perspective and at the same time act as a risk management warehouse is simply not feasible," says Patel. "In the end, transaction data models and risk management data models should be designed to support their respective requirements," he says. Nevertheless, he does believe "the next generation of technology, namely the financial server, will eventually span the entire range of financial instruments primarily because the vendor merely provides a framework and the user has the ability to fill in the workflow pieces. Net result -- a complete solution, to the user's specs, in record time."

SPECIAL FEATURES: other; illustration
COMPANY NAMES: Financial CAD Corp.--Product introduction
DESCRIPTORS: Software Product Introduction; CAD Software
PRODUCT/INDUSTRY NAMES: 7372431 (CAD/CAM/CIM/CAE Software)
SIC CODES: 7372 Prepackaged software
TRADE NAMES: FinancialCAD Server (CAD software)--Product introduction
FILE SEGMENT: CD File 275

4/9/12 (Item 2 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)

01634958 SUPPLIER NUMBER: 15042128 (THIS IS THE FULL TEXT)
Grasping objects: get ready for the upheaval. (financial software benefits from **object-oriented** technology) (includes related article on C++ and SmallTalk) (Cover Story)
Smith, Carrie R.
Wall Street & Technology, v11, n7, p24(5)
Dec, 1993
DOCUMENT TYPE: Cover Story ISSN: 1060-989X LANGUAGE: ENGLISH
RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 3417 LINE COUNT: 00270

ABSTRACT: Wall Street systems developers consider the benefits of **object-oriented** programming (OOP) technology in financial software. OOP enables programmers to code programs faster, and maintenance costs as well as the number of program errors that traditionally plague financial software are decreased. The innovation of financial instruments, particularly customized derivatives, is something with which only OOP is capable of keeping pace, according to experts. The properties of financial instruments can be encapsulated in discrete objects, and trading firms can use these objects repeatedly. Features can easily be added and taken out to create instruments that give a firm the competitive edge, and the use of objects lessens the occurrence of bugs. The up-front investment in OOP technology and its substantial learning curve are keeping many in the financial industry at bay, however.

TEXT:

Object-oriented programming is emerging as the technology capable of going head-to-head with Wall Street's complex financial instruments. But an intimidating learning curve, steep development costs and a newfangled way of looking at financial products may be scaring off many would-be users.

Roughly 450 Wall Street systems developers packed an auditorium at Metropolitan Life on Oct. 5 to learn the ins and outs of **object oriented** (OO) programming. Organizers had to change locations twice to accommodate the overwhelming response from financial houses either implementing or investigating this technology.

Nearly every major commericial bank, investment bank and brokerage firm sent developers who support front-office systems, with the largest contingents coming from Citibank, Chase Manhattan and Salomon Brothers. Others came from Smith Barney Shearson, Swiss Bank Corp., the Federal Reserve Bank, Moody's Investor Services, Standard & Poors, Reuters, Quotron and Dow Jones Telerate.

Not only was this the first meeting of OONY--the brand new users' group for **object-oriented** technology in New York--but it was also a change to hear Adele Goldberg, co-founder to ParcPlace Systems, a sping-off of Xerox Corp.'s Palo Alto Research Center, discuss object-based project management.

This is a technology taking Wall Street by storm.

Benefits are the ability to code programs faster, lower maintenance costs and reduce the number of bugs that creep into financial software. Industry experts claim that OOO is the only technology capable of keeping up with the increasingly speedy innovation of financial instruments, especially customized derivatives. "Objects are going to enable the financial community to get their hands around the complexity [of financial instruments] for the first time," says Patti Dock, a consultant and president of Pillar Systems Inc. in Sandy Hook, Conn.

That's because OO eases the process of dealing with rapidly evolving financial instruments by allowing the programmer to build an object library from which he can pull reusable chucks of code. Rather than perpetuate the tradition of coding each and every procedure from scratch, trading firms can encapsulate the properties of **financial instrument** components in discrete objects, which they can use over and over again.

Beyond the technical wizardry, can trading houses make money with OO? Yes, contend the software experts. "Some of the most creative uses of **object oriented** technology are on the trading floor," says Susan Cohen, a senior analyst at Forrester Research in Cambridge, Mass. For one, the ability to rapidly strip in and strip out features to create exotic instruments increases a firmhs competitive edge likewise its profit margin. Secondly, once an object has been created and "shelved" in an object library, its reuse should lead to a reduction of errors or "bugs" in comparison to traditional methods. And fewer bugs should translate into lower maintenance costs.

In addition, OO is revolutionizing the role traders play in the design of new products and in the tracking of product lifecycles. Instead of waiting for programmers to design financial products, some traders can assemble new hedging strategies by simply pointing to objects labeled Interest Rates, Yield Curve, Cash Flows and Currencies. One the technology has truly taken hold, expers see the chance for the integration between the front- and back-office functions. As functions normally associated with back-office work move onto the workstations, traders will take on a greater understanding of the full trading process.

But for all of the splendor of OO there are several hurdles the financial industry must still face. Beyond the onerous learning curve, which experts claim is scaring off potential users, OO requires a hefty up-front multimillion dollar investment as well as a modification of the way programmers traditionally approach projects. All this takes place in a relatively conservative industry comprised of traders who expect instaneous results--whereas experts estimate OO users can wait anywhere from one to two years for anything tangible.

Previously, Wall Street churned out new products despite serious lags

in technological advancements. Because firms were unable to process the most exotic products, the effects of the disparity could be felt in Wall Street pocketbooks. "One of the most intractable problems for firms is the length of time it takes to develop applications," says Cohen at Forrester. "that gets in the way of firms competing effectively."

To compete in the cutthroat world of custom derivatives, dealers must price, hedge and design transactions before software exists in order to book the business. "The standard development techniques cannot keep up with the way the market is evolving," says Jim Rogers, vice president of Sanwa Financial Prodcts (SFP) in New York. SFP, the three-year-old fixed income derivatives trading subsidiary of Sanwa Bank, was in a unique position in the industry when approaching OO. As a newer entity, the firm began to explore OO "from the beginning," says Rogers.

Rogers admits SFP was hesitant to dive head first into OO due to the industry's lack of familarity and dearth of knowledgeable programmers. But their need for effective technology was stronger. "With derivatives you cannot create a system and walk away because by the time you design and implement it, it is out of date," says Rogers, who chose Montage, object-based software from International Financial Technology (Indinity) Inc. based in Mountain View, Calif.

Today, almost two years after conducting a pilot project and deciding to go ahead with Sun Microsystemhs C++ system in their derivatives trading environment, SFP has survived the onerous learning curve and has laid the groundwork for an OO environment. Rogers goes so far as to say OO gave SFP the opportunity to jump into the market faster. "Once you build your infrastructure, it allows you to innovate quicker," he says. "The derivatives market is constantly changing, and that is where the **object oriented** market has tremendous value."

Role Reversals. Along with a decrease in the turnaround time necessawry for systems development, OO is affecting the traditional roles held by programmers. Programmers, who traditionally progressed up the corporate hierarchy and were compensated according to their project functions, are now considered "designers" and "implementers" in the new age of OO. And the differentiation between the two is somewhat sticky for human resources departments.

"The differentiation lines tend to go away," says Dock at Pillar. Dock advises Fortune 50 companies on their strategies, organizational structure, roles and the eventual price tags in making the transition to OO. Dock attributes this change to the nature of the beast--OO demands that all players in a project have a hand in all stages of development rather thabn working around a multitiered programming structure.

While causing upheaval in programming roles, OO is also revolutionizing the way members of the financial industry traditionally view the makeup of financial instruments. "From the technological perspective, **object oriented** technology changed the way we think about design," says Michael Packer, managing director of Bankers Trust. "It is a set of interrelated components rather than information that flows from box to box."

bankers Trust is currently utilizing OO in a joint venture with International Business Machines to build the LS2 system, a real-time commercial loan system. OO-based LS2 tracks the early discussion between the borrowers and lenders, the information of deals, the primary syndication of the deals, as well as the automated funding of loans, servicing, secondary marketing and the trading of the loans, says Rich Freyberg, managing director of loan system development at Bankers Trust. Due to the fact that Bankers Trust is currently reengineering their commercial lending process, "OO allows us to be flexible in the reengineering approach," Freyberg says.

OO is also prompting a change in trader functions. While there is still uncertainty as to the ultimate impact OO will have on traders, the revolution in software may spur a change for traders by placing traditional back-office processes on the workstation. "OO has empowered the traders at the desks to do things that they used to have other people do for them," Dock at Pillar.

Similarly, OO is prompting an evolution of the traditional trading

environment, says Cedric Packham, vice president and director of information services at ScotoaMcLeod in Toronto. "**Object oriented** technology is allowing us to create some very effective front-office systems," he says. Packham cites the capital markets trading desk Scotia implemented through Decision Software Inc. (DSI) of New York as an example. While Packham declines to attribute the move from back to front office solely to OO, he acknowledges that "some of the functionality being provided in the front office, such as real time exposure and position management, is moving work from the back office forward." Scotia's interest in OO led the firm to choose DSTS, DSI's real-time, multi-currency trading, position and risk management system. "We were looking for a trading system that could take us through the mid-'90s," says Packham.

While traders begin to reap the benefits of OO and create their own products via their workstations, a prospect that Philip Meese, director of technical services at Mercury Technologies in New York places "in the long-term," many questions emerge. What will happen to product innovation on Wall Street? "I suspect you are going to see a lot more innovative products," says Ron Dembo, CEO of Algorithmics in Toronto. And will traders require increased mathematical competency? Not so, according to Meese. Essentially, "you can map the traders' world out," says Meese. If OO keeps progressing, traders may eventually be able to develop highly structured deals based on elements such as cash, flow, date series, yield curves and underlying instruments with which they are already familiar.

But Michel Hanet, vice president of trading systems at Chase Manhattan, is a bit more skeptical about the prospect of traders taking a more active role in the actual programming of new instruments. "Traders won't develop instruments," he says, "unless it is layered in a user-friendly fashion." Chase took on OO three years ago when it purchased Opus by Renaissance Software of Los Altos, Calif. for interest rate derivatives. Though the bank is keeping Opus for the foreseeable future to handle all of Chase's global risk management products, Hanet says the bank is now starting to use VisualWorks, the latest Smalltalk product by ParcPlace Systems Inc. in Sunnyvale, Calif. According to Hanet, VisualWorks will be usable for the whole product spectrum at Chase. As an indication of the learning curve involved in OOO, Hanet says that though the latest system was introduced to Chase five months ago, the firm is now "barely seeing the changes."

As for the rumors regarding the power of OO to actually erase the more extreme distinctions between front- and back-office functions, Dembo at Algorithmics tends to believed otherwise. "I don't think it is integrating the front and the back office necessarily," Dembo says. "No one has really developed a good back office that is **object oriented**," he points out. Nevertheless, Mercury's Meese claims sights for the future are set on the tasks of front-office and back-office integration.

Weighty Drawbacks. For all of its purported miracles, OO does have rather weighty drawbacks. For one, the prevailing opinioin in the industry is that taking on OO means taking on a daunting learning curve--one that may have frightened away more than a few potential object users. "Is the learning curve driving firms away?" ask Forrester's Cohen. "Absolutely," she says.

Experts estimate is takes anywhere between one to two solid years for programmers to successfully learn the ins and outs of OO. The same experts estimate the learning curve for fourth-generation languages (4GL) or C is six months. But, according to Dock at Pillar, in that one- to two-year period a programmer learns a lot more about the system than he would about 4GL or C--including design, debugging and testing.

But for all the stress and strain involved in learning OO, experts claim the end results for outweigh any problematic roadblocks. "If the goal of truly reusable class libraries can be achieved, it is definitely worth the learning curve," says Packham at Scotia. Dock concurs. "If a person truly wants to learn **object oriented** technology, the learning curve is surmountable," she says.

And, apparently, traditional programmers are on equal footing with those entering the market fresh when it comes to learning OO. The deciding factor is the individual's ability to grasp the new approach to

programming. "In the past, software was developed in a divide-and-conquer fashion," says Cohen [redacted] Forrester. "**Object oriented** t[redacted]nology is a more holistic approach."

Dockp also pegs success in learning OO to the individual's attitude toward new technologies. When she conducted programmer training, "success depended upon whether I dragged them in kicking and screaming or whether they came in on their own," Dock says.

Scotia had a double-edged encounter with the OO learning curve, says Packham. In one sense, the firm did not experience the learning curve because DSI brought the OO-based trading system to the firm in production form, which eliminated the need for rigorous programmer training. But, to provide the firm with a better appreciation of the learning curve, two of Scotia's top programmers worked with DSI in the development stages. While one Scotia programmer readily grasped the concept of OO technology, the other programmer experienced difficulty. "This experience showed us that the individuals' background and experience played a significant part in their ability to grasp the concepts," says Packham. "Programmers experienced in user interface more easily adapted to the OO environment."

While the need for technological aggressiveness in learning OO programming is inarguable, Ray Dodd, a principle at Fusion Systems in New York, chooses to downplay the fear factor of the learning curve. "There has been a tendency in the past to present **object oriented** technology as a new mystical technology where you have to forget everything that you have learned," Dodd says. "That is clearly bogus." OO is based on previous platforms, knowledge and ways of doing things, Dodd says.

In addition to the arduous learning curve, a lack of technological OO experience in the financial industry and reportedly immature support tools have also raised concerns. "A lot of people are not ready to go for it," says Hanet at Chase. "It is a major paradigm shift." This reluctance has narrowed the field of experienced programmers--a necessary component for introducing the technology to a firm. The concern is rooted in "the depth or lack thereof in the talent pool," says Packer at Bankers Trust.

Out with the Old. Trying expectations for new technologies to the traits of old technology development is causing additional problems for OO's acceptance. First, traders are accustomed to tangible progress during product development. With OO "you spend a lot of lead time developing infrastructure with very little to show," says Meese at Mercury. "Traders are used to having something to show." But, again, experts harken back to the "no pain, no gain" theory. "One of the presumed benefits of **object oriented** technology is that if you do your homework upfront, you get a lot of benefits down the road," says Dock at Pillar.

Firms must be patient and realize that the task of building an object library alone adds to the task. "Until you get the objects on the shelf, you don't have the full benefits of the technology," says SFP's Rogers. "In developing objects you are paying your dues up-front and investing in technology." Meese at Mercury agrees. "There is an investment in infranstructure but once the infrastructure is there the applications are rapidly developed," he says.

Second, Dock prefers to play up OO's role in empowering individuals in the development process, whereas tools were about removing some of the more mundane aspects of human involvement. "It is not about control, and CASE [computer-aided software engineering] tools were about control," she says. Dock's concern is the industry's search for tools similar to those for older technologies--tools that may never exist. Regardless, Dembo at Algorithmics says even tools such as compilers for OO still have a long way to go. "The basic tools are still flaky," he says.

Overall, budgeting time and money for conversion to OO is a function of firm size and what applications the firm is seeking to develop, says Hanet at Chase. Financial investment in OO depends on several factors, including whether firms wish to convert all processes to OO or just specific projects and whether firms bring in outside consultants for the transition, says Dock. At the least, large corporations will need to invest millions, "not hundreds of thousands," each year in OO, she adds.

Hypothetically, if a firm is conducting three pilot projects, Dock estimates that the firm could spend in the neighborhood of $180,000 on

consultants as well as another $54,000 training internal personnel to be "mentors" on later projects. The other possibility is for a firm to send select internal personnel to an off-site apprenticeship program, which Dock estimates could cost $80,000 per project team. This is all in addition to training for the remaining internal personnel to acclimate them to OO as well as to software costs, which generally depend on a firm's deal with a vendor. "Every firm has very different and personalized goals and objectives in mind when transitioning to objects," says Dock. Regardless, after the first year a firm's OO costs will "absolutely decrease," says Dock, because the first year is spent training people.

So where will OO go from here? "OO is still coming of age," says SFP's Rogers. Likewise, Packer at Bankers Trust says there is "a tremendous amount of evolution to come" with regard to OO. But there is the inevitable fact that all technologies eventually gather dust. Despite his declaration that "**object oriented** technology is a pretty big evolutionary step," Dembo at Algorithmics remains steadfastly realistic about the furture. "Within five years you will see another paradigm or some major enhancement," he says.

But for now things are looking up for OO. Says Dr. Jeffrey McIver, director of financial engineering at Infinity: "**Object oriented** technology won't be the new kid on the block. It will be the only kid."

Related article: AT THE HEAD OF THE CLASS . . .

Here are the top two **object oriented** (OO) languages wrestling for Wall Street's attention:

C++: Developed by Bjarne Stroustrup at AT&T Bell Laboratories as an add-on to its C language, C++ is virtually synonymous with OO. C++ is referred to as a "hydrid" language, as it is a combination of two programming approaches--the traditional method represented by C and pure OO approaches represented by Smalltalk. C++ is often thought of as the best choice for those familiar with C.

A conceivable drawback to C++ is the financial industry's perception that it is more of a transition language than true OO. "C++ is stil a revision of C and not totally **object oriented**," says Jim Rogers, vice president of Sanwa Financial Products (SFP) in New York. SFP uses Sun Microsystem's C++.

Smalltalk: Developed by Alan Kay and a team of researches at Xerox's Palo Alto research center, Smalltalk is not a spinoff of an earlier language and is therefore considered a more "pure version of OO technology--pure OO hints of greater extensibility to those in the marketplace. Smalltalk is considered more suitable for previous users of Cobol. Smalltalk's biggest battle will be marketing itself to Wall Street users of C--a healthy portion at the least.

SPECIAL FEATURES: illustration; photograph
DESCRIPTORS: **Object-Oriented** Programming; Trends; Financial Software; Programming Language
FILE SEGMENT: CD File 275

4/9/13 (Item 3 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)

01616922 SUPPLIER NUMBER: 14330777 (THIS IS THE FULL TEXT)
Prometheus unplugged. (wireless communication is poised to revolutionize financial trading) (Forbes ASAP: A Technology Supplement)(Quick Studies)
Young, Jeffrey
Forbes, v152, n6, pS149(2)
Sept 13, 1993
ISSN: 0015-6914 LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 1334 LINE COUNT: 00101

ABSTRACT: Swiss Bank Corp is aggressively pursuing the application of wireless computing and communications technology despite its history as one of the most conservative of Swiss banks. The company has supplemented its

array of high-end workstations with various wireless communications devices such as radio pagers and palmtop computers. Dwight Koop, one of the bank's information technology executives, hopes eventually to have the bank's commodity traders feed prices and sales information by handheld computers that they could carry on the floor of the Chicago Board of Trade. Swiss Bank is also in the process of completing its acquisition of O'Connor Partnerships, a Chicago trading operation that uses sophisticated software to manage trading in financial derivatives, such as options and index instruments.

TEXT:

Four floors above the Chicago Board of Trade's commodities futures pits lies trading's real future.

DWIGHT KOOP WORKS for one of the world's largest, and most conservative, Swiss banks. As he attends a meeting in an elegant conference room, the beep from the pair of palm-sized black boxes on the table in front of him is barely audible. Nestled in a leather carrying case and held in place by Velcro fasteners, the Hewlett Packard 100LX, a top-of-the-line programmable calculator-cum-palmtop-computer, is tethered to an Ericsson GE Mobidem wireless modem whose flexible plastic antenna is waving in the breeze from the air conditioning. Both devices are battery-powered. A symbol atop the modem's LCD screen flashes on and off every few seconds, indicating that something is coming in, or going out.

That's not the only thing going on.

In the meeting room, the voice of an executive in Switzerland booms out of the speakerphone. Beyond the glass wall, a mezzanine gallery circles a basketball-court-sized three-level space filled with pods of desks, stacks of computer monitors and the requisite wall of moving ticker symbols, stock prices and news feeds. This is the trading floor of the Swiss Bank Corp. in Chicago, four floors above the chaos of the Chicago Board of Trade's commodities futures pits. Here, in contrast to the frenzy of colored jackets and hand gestures in the pits, all the action happens on Next, Sun Microsystems, Hewlett Packard, Symbolics and other workstations--as well as on a growing number of handheld, wireless devices of all kinds.

This afternoon the group in the meeting room is discussing new software developments that Swiss Bank is planning. But Dwight Koop, 45 years old and executive director of information technology for Swiss Bank Corp., has other things on his mind. A few taps on the tiny keypad move him through a list of 15 E-mail messages that have landed in his wireless mail-box with the latest beep. The HP 100 and modem are small and unobtrusive enough that he can use them without disrupting the meeting. He stops at one message, pulls it up on the screen and squints at it through his glasses. A faintly worried look passes over his face. A few more taps and a reply shoots out into the ether. Simultaneously, his SkyTel Sky-Word beeper starts vibrating. Koop checks it, then punches his watch, a Dick Tracy-like digital artifact that not only tells the time but also is a paging device. He stands up, closes his equipment, mumbles something about "putting out a fire" under his breath so the telecommuting executive can't quite hear him over the speakerphone, and breaks away from the meeting.

Koop loves the gadgets. But he's also charged with integrating them into Swiss Bank's work culture, and that, he says, won't be easy: "It's not all ready for prime time yet. The service's interface and interaction model is troubling. For someone who has spent years messing with computers, it can be navigated. Far too often I have to fight it [the system] to make it work."

Koops says wireless trading will "languish" until somebody figures out all the pieces and integrates them. Today Swiss Bank sends its traders onto the floors of the exchanges with print-outs from its internal computer systems. The bank would like to someday equip its traders with handheld machines. "It won't be anytime soon," Koop says. "Trading is a contact sport."

Koop's job involves keeping the bank's global trading systems up and running. These are not just any trading systems. Over the past few years, Swiss Bank has acquired the majority of Chicago's O'Connor Partnerships (the final portion of the acquisition is awaiting SEC approval). O'Connor

is one of the most highly secretive, technically sophisticated and profitable of the world's trading operations dealing in financial derivatives--products like options and index instruments that are derived from underlying traded securities such as stocks, bonds and currencies. A leader in the application of mathematical algorithms to options, currency and **financial instrument** trading, O'Connor was once known for trying to avoid all publicity. It once shredded the packaging materials for the workstations it bought and required all employees to sign extensive secrecy agreements. Even today the closest a visitor can get to an O'Connor computer screen is the gallery, a good 30 feet from the trading floor.

Creating new bundles of instruments and options to sell in response to customer requests--instantly--is a growth opportunity that savvy traders have been eager to exploit. All of this is much better handled by computers, which O'Connor realized in the mid-1980s when it led the financial community in moving first to Sun machines and later to more sophisticated workstations. But the new trader's edge is literally up in the air.

WALKING AROUND--GLOBALLY

If any trading company figures out wireless, it probably will be Swiss. Swiss Bank likes to push the technolo-envelope, for example, buying 500 Next machines a few years ago because it believed in the future of **object-oriented** development systems. "We have no idea if Next will make it in the long run," says Koop's boss, Craig Heimark, Swiss Bank's managing director of technology. "But someone will succeed with objects. And we'll already have a wealth of experience programming in this kind of environment."

Koop enjoys explaining that he has signed 425 nondisclosure agreements with technology companies. "When companies ask if they can qualify us as a beta site," he says, "I explain that they don't seem to understand. We qualify them." This is real money being handled--security is a big issue. "Dial-in modems simply don't exist in our world," he adds quietly.

For all of O'Connor's history of secrecy, Swiss Bank is talking about moving its proprietary risk-management trading systems to customer sites. The idea is for the company to let customers manipulate versions of its proprietary analysis tools, then sell them the financial products that meet their needs. Enter wireless. "We want our managers to work by walking around--globally," Koop explains. "But we sure don't want these top-level people fumbling for the phone jack and negotiating with the PBX operator to get a dial-out line so they can check the day's market."

For all his complaining, Koop is still certain that wireless will be a key communications component over the next few years. His own love of the toys convinces him.

On another day Koop is sitting in a hotel in Sausalito, at a window over-looking San Francisco Bay and the city skyline. His computer and modem are open in front of him, and the radio signal is strong. The indicator is flashing furiously. His attention is intermittently pulled to the screen. It is more than a little disconcerting to try to talk with him. We may have to learn a new social dynamic in the era of ubiquitous wireless machinery.

Suddenly he chuckles, and pushes the unit over to display a message he's just received. "A bunch of us have an unofficial contest to send a message from the most unusual location," he says. "Have you ever heard of this place?" The message is tough to see in the glare, but when the tiny LCD display is positioned correctly, it reads: "Do I win the contest? I'm sitting at the bar of the Jaguar Club in Atlanta." (The Jaguar Club is one of the more infamous topless bars in the South.) Ah, brave new world. As it turned out, that message didn't win. The winning one was sent from atop Elvis' grave at Graceland.

SPECIAL FEATURES: illustration; photograph
COMPANY NAMES: Swiss Bank Corp. (Switzerland)--Communication systems
DESCRIPTORS: Wireless Network; Financial Services; Futures; Commodities; Brokerage Industry
FILE SEGMENT: MI File 47

01360727 SUPPLIER NUMBER: 08411954 (THIS IS THE FULL TEXT)
Early adopters report architecture has pluses; but technical challenges remain formidable. (client-server architectures)
Braasch, Bill
Software Magazine, v10, n5, p63(6)
April, 1990
ISSN: 0897-8085 LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 3835 LINE COUNT: 00313

ABSTRACT: Early users of client/server computing architectures report significant benefits, but tools for building client/server applications are just being developed, and some connectivity and control issues are still being worked out. The client/server concept separates applications into three components (user interface, application logic and server), enabling selection of the best technology for each. Advantages of the architecture include off-loading of processing to less-expensive hardware, greater responsiveness to user needs, reduced training time, shared resources over a network and more robust applications. Lower workstation costs, more powerful microprocessors and new software capabilities are helping to drive client/server computing growth. Several examples of the use of client/server applications are described. The need for interoperability and control standards is discussed.

TEXT:

EARLY ADOPTERS REPORT ARCHITECTURE HAS PLUSES

In the film Star Wars, Luke Skywalker had all the advantages of client/server computing. His servers, R2D2 and C3PO, were there when he needed them. With them around to keep him out of trouble, Luke could concentrate on the war against Darth Vader. He did not have to know how R2D2 and C3PO worked, or how to make them communicate with each other. They took care of that themselves. And none of the wires were showing.

But that was the future. This is 1990--the year that the first significant steps toward client/server computing are taking place. Dropping workstation costs, more powerful processors and new software capabilities are bringing new opportunities to the desktop, via client/server computing.

The driving force behind the move is the applications. Customers are identifying applications that put a lot more data on the screen. These applications call for bit-mapped terminals with graphical user interfaces and database managers to handle the data.

A look at the early adopters of client/server computing reveals that the technology is, indeed, taking hold. Moreover, the evidence suggests client/server computing is making good on promises. Client/server systems are not only creating significant benefits by off-loading processing to less expensive platforms, but they are also demonstrating greater responsiveness to user needs.

In fact, some corporations, while admitting they have implemented client/server applications, were reluctant to divulge details because they are using these applications to gain competitive advantage in their markets.

Client/server architecture is expected to be in full bloom within the next five years. But while there are compelling business reasons to implement client/server applications, the tools for building these applications, as well as connectivity and control issues, are just now being worked out. There are still a few wires showing.

Client/server computing first took the form of shared printers, file servers and electronic mail on LANs. As technology has progressed, so has client/server. At the client end, user interface technology has advanced dramatically. At the server end, we now have databases, knowledge and inference servers, as well as attachable processing resources.

Classical mainframe applications have used dumb terminals for years to move data between the user and the database, treating the application as a monolith. Client/server computing separates an application into three pieces: the presentation layer (or the user interface), the application logic and the server (database, mail, fax, etc.). By separating an application into these layers, the best technology for each can be chosen.

Client/server computing takes advantage of advances in user interface technology, relational database capabilities and network connectivity. Client/server applications present a single image to the user--regardless of the complexities under the hood.

But client/server computing is more than just a new face on an old application, or a shared database behind a personal computer. It is a new way of thinking about the way we use information technology.

"When I meet with a client to discuss implementing client/server architecture, I first ask what the business opportunities are," said Dr. Robert S. Epstein, executive vice president of Sybase, Emeryville, Calif. "We're there to solve business problems. No one moves to a new technology unless they are solving a compelling business problem with that technology."

Epstein finds two driving business reasons, both tied to gaining competitive advantage: adding value to a product or service using information technology, and reducing the time to market for products and services.

REDUCED TRAINING TIME

One example of the business benefits of client/server computing is the use of improved user interfaces to reduce training and support costs. This can be seen in how AMR Travel of Carrollton, Texas, is using InFront/DS from Multi Soft, Inc., Lawrenceville, N.J.

Multi Soft offers a fourth-generation language for developing client services in front of existing CICS applications. The company's InFront/DS development system includes a screen management system, a programming language and a local database.

Customers use these products to improve the user interface and perform editing on the workstation, according to Ross Altman, Multi Soft's vice president of marketing. This dramatically reduces the learning curve at the workstation, he said.

Multi Soft also offers EasySAA, whic expedites development of CUA (Common User Access)-compatible interfaces to existing applications.

Doug Brown, director of technical services for AMR Travel, used Multi Soft's tools to develop a user interface for a sales and marketing system, while keeping the data on the mainframe database. At the client side are networked IBM PS/2s, with an IBM MPA board providing a gateway to the host. Two servers, both IDMS databases, are available: the reservations database, and the sales and marketing database.

With the personal computer interface, users can pick codes from pop-up list, reducing the learning curve. "With the editing at the workstation, we have a much better ability to edit data, since the personal computer lets us edit each field as it is entered," Brown said.

AMR's system runs on MS-DOS, with extended memory. Brown said he is content with the DOS configuration and sees no compelling business reason to go to OS/2 at this time. "Even though there are some nice OS/2 applications available, our users need to be able to mix DOS and OS/2 applications."

Another company that reduced training time using client/server computing is Marine Midland Bank of New York. Ken Dubensky, vice president, said, "It used to take a commercial loan person six months to become productive. By adding a new front end to a system that's 15 or 20 years old, we expect to reduce that time to two months."

Dubensky used Easel from Interactive Images, Woburn, Mass. Easel supports development of the user interface, as well as a C language programming environment installed at Marine Midland. The new user interface collects data for the mainframe-based commercial loan system and two local dBase databases the bank developed in recent years. Since all the data is collected in a single-user session, consistency is guaranteed.

Based on the results he is seeing with the commercial load system,

Dubensky expects the bank to broaden its use of client/server applications.

CHANGES IN USE

Aspen Research, Inc. of Burlingame, Calif., also offers a fourth-generation language for developing client services. Aspen designed its product, Mozart, so its customers could quickly and easily upgrade the look and feel of existing mainframe applications--without rewriting any code. According to Alan Parnass, president and founder, "Because of the high costs and risks, it is unlikely that mainframe applications will be replaced. MIS is simply too busy adding functionality to go back and rearchitect."

Parnass also sees a change in the nature of the user. "We are moving from the data-entry user to the professional transaction worker, a level above the see-and-key types." Mozart offers an evolutionary approach, according to Parnass. "Professional workers gain direct access to mainframe applications, but they can operate them without formal training."

One user of Mozart is Immanuel Medical Center in Omaha, Neb. "Or physicians needed online access to patient information, but they resisted the log-on process and input screens of the online system," said Dan Connolly, systems and programming manager for the medical center.

"With Mozart, we were able to create a new front end. Now they use a bar code scanner to enter their code. The system responds with a display of the patient list and a laser-printed report. They don't even have to tough the keyboard," Connolly said.

Another feature of Mozart is its support for up to four concurrent sessions across IBM and DEC processors. Mozart supports a character mode user interface within DOS' 640K limit. The extended memory version supports a graphical user interface. The OS/2 version will support Presentation Manager.

OFF-LOADING DEVELOPMENT TO PCs

Micro Focus, Palo Alto, Calif., offers the Cobol/2 Workbench and other PC workstation software tools that can be used to off-load development of applications to the personal computer. Micro Focus' argument in favor of Cobol is that the language is widely used and is portable to a number of platforms. Applications built with these tools can be run on the mainframe, on a LAN or as client/server applications.

According to Karl Kramme, product manager, Cobol/2 Workbench includes a Diaglog System which is a runtime controller of the user interface. It includes tools for generating a character-based interface. Preprocessors are available for using third-party database tools, such as XDB from XDB Systems, College Park, Md., and SQLBase from Gupta Technologies, Inc., Menlo Park, Calif., for developing DB2 applications.

Micro Focus also offers a development environment for running CICS under OS/2. Support for OS/2 Extended Edition, including its Data Manager database component, is built into the compiler, according to Peter Katz, technical marketing manager.

Barbara Bouldin, manager of third-party relations for Micro Focus, said, "The technology is here now, but an attitude of pioneering still exists. The tools are still in the hands of the architects and designers. The next step is broad-based use."

From a performance standpoint, client/server architecture allows traffic to be off-loaded from the transaction processing network. For example, American Airlines, Fort Worth, Texas, has moved its yield management application, called Interactive Dinamo, to a local area network. "With OS/2's multitasking, our users can work in one part of the screen while a report is being developed in another window," said Kennon Grose, acting manager, new technology systems planning/development.

The yield management system provides critical information to flight analysts who try to ensure that every flight leaves the gate filled to capacity, and with the highest fare that the market will hold paid by the passengers. At the same time, the analyst tries to avoid overbooking. American Airlines claims it has the lowest rate of denied boarding of all U.S. carriers, and it credits the yield management system for that success.

"We standardized on OS/2 and Presentation Manager over a year ago," Grose said. "We have six mission-critical applications in production or in development on the platform."

Some of the more advanced work introduces knowledge in the form of inference into the application. American's QMAX is an application that handles booking requests for large parties. QMAX includes rules that apply by class of traveler.

American Airlines uses the SQL Server engine developed by Sybase to manage a 1.2G-byte database on the local area network. Although early applications were coded using Presentation Manager and C, American has begun to use Choreographer from GUIdance Technologies, Inc., Pittsburgh, Pa., to develop a user interface. Choreographer is an **object-oriented** environment for building graphical user interfaces.

"The first screens pay the cost of building the library of reusable objects, but once these are in place, additional screens can be developed very quickly," according to Ken Harkness, president of GUIdance Technologies.

Grose agreed: "With Choreographer, we can develop rapid prototypes. If we want to replace Choreographer code with C for performance, we can easily do that."

However, Sybase's Epstein said that true client/server applications require a new way of thinking about applications, which is more than just putting a new user interface on an existing application.

One of Sybase's customers is building a client/server application to sell tickets to sporting and entertainment events. The application tells the user what seats are available, then reserves the seats, calls Visa to charge the ticket and then prints the ticket. Many of Epstein's customers, like this one, invoke multiple servers from a single client interaction.

Epstein expects the reusability of servers, such as the Visa server in the example above, to reduce the time needed to build applications, reducing time to market for service industries like banking. "Imagine the advantage a financial institution would have if it could quickly introduce a new **financial instrument** to the market," noted Epstein.

Client/server computing also enables a new class of applications. Analyst Esther Dyson, who is also publisher and editor of the newsletter Release 1.0, New York, has popularized the term "groupware" for software that allows groups to collaborate toward a work product.

An example of such an application is Lotus Notes from Lotus Development Corp., Cambridge, Mass., which came to market with a great deal of attention. Originally previewed at Dyson's 1988 PC Forum, Notes provides an environment for sharing text and graphic information, developed internally or gathered form external sources. Notes also advances the support for collaborative work by allowing for information sharing within and across LANs.

One user of Notes is Price Waterhouse of New York. According to Sheldon Laube, national director of information and technology, Price Waterhouse was looking for a way to manage its expertise. "As consultants, that's our stock and trade," Laube said. Price Waterhouse communicates through phone calls, faxes and Federal Express. Laube was looking at E-mail packages when he came across Notes.

"With Notes, we see an opportunity to let our professionals collaborate on a global basis," he said. Price Waterhouse has introduced Notes to support several of its internal groups, and will eventually install 10,000 copies of the product.

The system manages forms and databases that have been imported. Users can browse for material, attach comments and route information to others. "People think it's terrific," Laube said. "It looks like any Windows application. Our people find it easy to use and they quickly find applications for it."

One application they use is the Reuters News Fed. As the news feed is received, Notes performs a keyword search and categorization, so that users can quickly find news of particular interest.

Laube feels that Lotus Notes justifies the cost of the dedicated OS/2 server, the client workstations (which were already in place) and the software, which is priced at $62,500. "It will help us to serve our clients better by giving our professionals better access to our professional information. We expect to gain a major competitive advantage by using Notes."

A large number of client/server applications will be built as business applications rather than installed as shrink-wrap software. In fact, Notes is really a programmable platform for group interaction, with forms, rules, triggers and mailboxes. According to Jim Manzi, president of Lotus, "Notes may be more of a service than a product," because the applications built on Notes will give it its value.

THE STANDARDS ISSUE

While the benefits of these client/server computing projects are clear, making the move to client/server is still a technical challenge. Client/server applications often require that components from different vendors interoperate. This brings the standards issue to the front lines.

Advocates of open computing architectures argue that there should be agreement on standards so that processors, applications and databases can be intermixed on the network, as called for by the application. To do this, the following standards would need to be agreed upon:

* Data must be accessible in a common language that does not require knowledge of how or where it is maintained.

* Different operating systems need to be able to work together to accomplish work.

* The network that ties these processors together should be hidden behind the user interface.

When it comes to standards, however, companies don't just run toward each other in slow motion with open arms like lovers in a Fellini movie.

The one standard that appears to have been settled upon is SQL. But, according to Umang Gupta, president of Gupta Technologies, Inc., of Menlo Park, Calif., "SQL is not SQL is not SQL. SQL syntax capability is fairly well understood--the problem is not the snytax, but the API [application program interface]." Without a common interface at the application level, programs that access DB2, for example, would need to be changed to read Oracle, for example.

Underlying the syntax is a mix of functionality. "Two-way browsing, an inherent requirement in a graphical user interface, is not currently available in DB2," Gupta said. So to hook up technologies without losing any of their advantages, Gupta has introduced SQL Windows, an application development tool that hides environmental differences. With this product, an application can be developed using **object-oriented** tools, or the code which has been generated by the tools can be retrieved from a window and modified.

According to Gupta, "The on-us for making this work together is on the front-end vendors to provide drivers and routers, or on the connectivity vendors to do this for them." Gupta claims that it is easier for a company like his, with tools for the front end and back end, to solve this problem for its customers.

Forecasts of the MS-DOS, OS/2 and Unix share of the desktop operating system market vary widely. Although none seem to have the ability to eliminate the others, there is movement toward interoperability of these environments.

In support of interoperability, some companies have created products that hide the differences. For example, Choreographer will generate a user interface for Windows or Presentation Manager. Of course, each proposed graphical user interface standard claims to have its own advantage. Tools that generate multiple interfaces, however, must operate at the lowest common denominator level, sacrificing some of the capabilities.

Oracle Corp. of Belmont, Calif., is planning to offer facilities to hide the network. According to Gene Shklar, director of marketing for Oracle's Network Products Division, users should be able to turn on the network in the same way that they turn on the personal computer today. The network should appear to users as a transparent set of services and resources.

"Enterprise computing is in the Heathkit stage today," Shklar said. Oracle is creating "Universal directory Service," a product that would talk to products such as Banyan's Street Talk, Sun's Yellow Pages, TCP/IP's Domain and DEC's Distributed Name Service. Also, a "Universal Authentication Service" will provide one log-on to the network. Based on this log-on, UAS would talk to the other servers on the user's behalf,

Shklar said.

Application developers need to be able to construct, test and install applications into a client/server configuration. This problem brings new concerns to the issues of library control and change management.

DECENTRALIZED DATA CENTER

One of the results of client/server computing is the decentralization of the data center. Alan Pearson of Bank of America, who is implementing banking applications using OS/2 workstations and DB2 server, said, "It's like having lots of little data centers out there. . . . What happens when you want to make a change, or diagnose a problem, or tune performance? You can't really call someone at 3 a.m. in a bank branch somewhere and ask them to reboot from the A drive. There's nobody home."

According to Pearson, "IBM's Repository and NetView products are envisioned as the information resource and management tools to support large-scale client/server environments."

Sybase's Epstein said that a client/server architecture must be built based on the potential that the clients will be loading their own software. "You have to construct the system assuming that the KGB is writing the application and the U.S. is writing the servers. The secret to making the whole thing work is making the servers smart enough so that the application can't trick them," he said. "We stress the importance of making the servers embody all the business logic to decide if it's a reasonable transaction or not."

The outlook as we move to client/server computing is that mainframes will become large file servers, as the users interface and processing logic move to the client platform. Mainframe environments have the networks in place needed to support client/server. These networks have driven dumb terminals in the past, but are increasingly routing messages from personal computers to the mainframe. So the move from a mainframe perspective is evolutionary in the direction of moving processing off the mainframe, but keeping the data centrally controlled.

"Data processing organizations are committing their resources to providing added functionality. There is no time to go back and rearchitect existing systems, especially when you can improve their value by moving processing to the client and continue to use the existing application as a server," said Aspen Research's Parnass.

While the technology still has some kinks, client/server computing is definitely moving forward. It seems that those who see it from the mainframe perspective feel that if users can be given a friendlier interface that is intuitive and self-teaching, then client/server is worthwhile.

ADDED VALUE

From the personal computer perspective, if client/server lets users share resources over the network, this is added value. From the user's point of view, client/server computing can provide more robust applications with multiple windows, multitasking at the workstation, collaborative work and seamless transition between internal applications like American's Interactive Dinamo and packaged applications like Microsoft Excel, running under OS/2.

According to Epstein, the applications requirements, not the technology, will drive the move to client/server computing. "People buy OS/2 because they need it for networking, not because it's new or discounted. The same is true for workstations and database technology. The applications requirements are driving the move toward client/server computing."

The first appilcation is often a departmental system: The users in a department need to be able to work with a lot of data at once, and the data needs to be presented in a format that makes it more comprehensible. This calls for a bit-mapped graphical user interface and a database manager.

Once the first application shows its benefits, others follow. "We are beginning to talk to customers about using our technology at the enterprise level," Epstein said. "As we show that we can provide increased capability on lower-cost platforms, we expect more applications to more toward this approach."

But the movement may be slow. In a recent study of the database

market by Sentry Market Research, Westborough, Mass. fewer than 20% had evaluated a client/server database product.

Bill Braasch is president and founder of Data Base Architects, Inc., a consulting firm located in Alameda, Calif.

CAPTIONS: Percent of large sites that have evaluated client/server systems: 1989. (graph); Products that support client/server computing. (table); Percent of firms that plan to use PC data bases to front-end host DBMS. (graph)

SPECIAL FEATURES: illustration; graph; table
DESCRIPTORS: Client/Server Architecture; Utilization; Products; Trends; Performance; Case Study; Computer Software Industry; Outlook
SIC CODES: 7372 Prepackaged software
FILE SEGMENT: CD File 275

4/9/15 (Item 5 from file: 275)
DIALOG(R)File 275:Gale Group Computer DB(TM)

01245627 SUPPLIER NUMBER: 06728259 (THIS IS THE FULL TEXT)
New directions in database management software.
Mazzella, Donald P.
Wall Street Computer Review, v5, n8, p73(6)
May, 1988
ISSN: 0738-4343 LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 3384 LINE COUNT: 00271

ABSTRACT: Database management systems (DBMSs) help users cope with increasing quantities of information from widely disparate sources; the packages can help financial services companies gain a competitive edge. Many DBMS products are available to meet the evolving needs of the financial services industry; they have become easier to install and use, offer improved price-performance, and can accommodate both text and financial data. Information about more than 30 such products, ranging in price from $50 to $75,000, is presented. Criteria that enter into the selection of a DBMS include its ability to handle information of varying degrees of complexity, the presence of functions for distributing more data to managers and workers, and data capacity. The outlook for the DBMS market is discussed.

TEXT:

New Directions in DATABASE MANAGEMENT SOFTWARE

Kenneth E. Lehrer serves as chairman of four banks in Louisiana at the behest of the Federal Home Loan Banks of Dallas. He also runs a thriving financial and real estte consulting business, Lehrer Associates, in Houston. Frequently, he is called as an expert witness in complicated trials involving real estate, mortgage, and **financial instrument** evaluation.

Because he has four degrees from New York University, Dr. Lehrer also is actively involved in the NYU Alumni Association, teaches at the University of Houston, and personally repairs his vintage Rolls Royce, scrounging the country for parts and knowledge.

Keeping track of all these efforts while operating between Dallas, Houston, and New Orleans is a challenge for himself and his staff. More importantly, much of the information he needs to track and keep detailed financial and analytical backup records of comes from widely disparate sources.

"I have to monitor the stock market, the mortgage-backed securities area, new federal banking regulations, as well as a host of other information sources," he relates. "More importantly, since I'm often called at the last minute by attorneys, I have to be able to bring information together quickly and easily. When I collect the inforamtion, I don't know where to stick it in a cubbyhole that I can retrieve quickly."

Lehrer's dilemma is one facing many financial and investment

managers. A self-proclaimed neat and precise person by nature, Lehrer notes he did not perceive that he had a problem until his business expanded. "I was satisfied with paper files, my Rolodex, and a word processor. My staff fed me any information either in written form or by diskette," he recalls. "That worked fine until 1986.

"Before 1986, my business was restricted to creating institutional presentations on individual construction projects in order to help builders obtain financing," he remarks. "Then I started to get requests for analyses of faltering projects. This required me to put together area population and building data and to tie it to national economic and equity information."

Working with local banks and securities firms, Lehrer's company became involved in doing equity evaluation, particularly when it was associated with real estate. "That's what brought me to the attention of the Federal Home Loan Bank in Dallas and tripled my information needs," he relates.

With his involvement in banking, Lehrer was forced to set up a whole new database and file management system. Once he completed this project, he found that his old and new files were incompatible.

Moreover, he found that information loaded in one file could not be accessed the same way the other files were. He couldn't move back and forth between various information sources and time was being lost slipping diskettes and programs in and out of his personal computers.

Because updated and cross-indexed information is critical to the success of his business, Lehrer sought a more free-form way of managing his information. He looked around for a solution and chose DayFlo Tracker from DayFlo Software, Irvine, Clif. "While not perfect, it gives me the ability to track textual, financial, and public information while easing my workload and report writing," he says.

As an added benefit Lehrer finds he can keep track of people in the U.S. who have the same make and year as his Rolls. "They're the ones who tip me to the best places to find parts. They also tell me how to fix the car. I throw their tips into the database as well."

Apart from his car repair concerns, the Houston financial entepreneur's concerns are similar to those of others in the brokerage and investment industry. As the impact of rapid changes in the financial community is felt, many people's job descriptions and workloads change. This often means changing the information usage patterns they had built into their personal computers.

Cindy Smith, an assistant vice president at Merrill Lynch's Capital Markets area in New York for five-plus years, was heavily involved in operations and client management. After participating in some of the marketing aspects of financial services, she began a job search in that area.

"I had my own PC-based data file using [Lotus] Manuscript," she says. "But I had to rado it all when I started my job search and I'll probably do it again when I start my next position."

Free-form File Structure

Smith is not alone in acknowledging the differing and often incompatible structure of database management systems. Personal computer users are anxiously awaiting the next generation of products that are more flexible in their file structure and screening algorithms.

Companies like Lotus, MicroSoft, BAC Software Inc., and DayFlo are rushing to meet the needs. Indeed, if a glut of new products appears on the market, the problem on Wall Street will be choosing the right product for individual use.

For heavy data users and network suppliers, the problems and inherent opportunities are even more complex. The volume of information is reaching the saturation point for many dat management systems. In the past, carefully crafted information analysis applications have been the difference between profit and loss for many financial services companies.

After four evolutionary stages, database management systems for the financial services industry are poised for a technological and sales expansion. From their initial usage as information managers, beyond flat-file management tools, through to relational databases and free-form or textual/numerical products, database management systems (DBMS) have

helped Wall Street cope with the information explosion of the past decade.

Huge profits have been wrung from the information explosion by companies anticipating and using new technology. With this year's releases, "databse technologies may be at a point where the new products will leapfrog current leading edge products," states F. Warren McFarlan, professor of computer science, at Harvard Business School.

Now, with so many DBMS offerings available and more being announced, investment managers, analysts, brokers, controllers, and just about anyone else in the financial services industry would appear to have enough choices.

Information Cascade

For many, the new technology is coming just in time. "Each day we're seeing new needs for information with less lag between creation and use," says Patrick Thomas, database consultant to Nomura Securities, New York. "The cascade of information which has bombarded Wall Street and other sectors of the financial services industry doesn't appear to be slackening."

Typical of the variety of uses is a system called Paradox, from ANSA Software, Belmont, Calif. "We're using Paradox for both single and network applications," says Kip Ryder-Lewis, vice president of the Information Center at Bear, Stearns & Co. Inc.

"One system is used to track clients, send letters to brokers, do pricing and reports, and to maintain a databank of textual information on particular equities," he relates.

In addition to client applications Control Plus, by Phase One Systems, San Leandro, Calif., is offering a relational database management system that incorporates query language, data dictionary, screen management functions, and a report generator.

Like many other products, its focus in recent years has been in eliminating as mcuh customized programming as possible. "We think that is what the industry is demanding--less customization--and products that are easier to install and use," says Ron Gibbs, president of Phase One Systems.

Interactive Technology Inc., Beaverton, Ore., plans to go a step further. According to its president, Roger Brown, the company's aim is "to allow users to build a complete customized application without them being a programmer. We recognize that clients want 'to do it their way' and that current software doesn't serve that need," he continues.

Brown, Gibbs and others view their clients' needs as growing geometrically and having the products taking more and more advantge of new technology. David Burns, group product manager for PFS: Professional File from Software Publishing Corp., Mountain View, Calif., finds the trend may be there but as he puts it, "structured, flat file systems are still satisfying the needs of almost 80 percent of the financil services users we surveyed late last year. We made a corporate decision to provide the best product in this area and we are always looking at what our clients are doing," he remarks.

Many industry managers point to the growing number of information companies who are positining themselves to offer both text and financial dat. "One sign of the times is the indexing of companies introduced last fall by the Wall Street Journal and New York Times," says Gene Duchin, president of Parliament Hill Corp. in New York. "When they start pumping out the data as combined text and financial numbers, Wall Street players are going to need a whole new set of products."

Besides coping with this anticipated torrent, Warren McFarlan says he believes industry leaders will need to find wys of taking advantage of the situation. As he points out, "the environment has become even more competitive" given the advent of 24-hour trading; greater amounts of vital, on-line textual dataf newer, more sophisticated information gathering techniques; and the growing understanding that minute-to-minute intelligence is key to long-term profits. In addition, there is a growing demand for software systems to simultaneously handle text and numericl data.

According to McFarlan, whose specialty is identifying ways for clients to use computers to gain competitive advantage, "the economies of technology are beginning to come into play on [Wall Street]. The companies

that learn to use computers to gain competitive leadership can use this period to extend their leads."

McFarlan goes on to explain that "many managers want to exchange software for people. It's inherent in the evolution of the industry which is information-based, the software being offered or developed and the growth of data, that significant technological changes are happening."

Very specifically, McFarlan points out that bigger organizations will gain a competitive edge over their smaller rivals because larger firms have the capital to invest in these new technologies.

"The smaller companies will be at a disadvantage trying to identify and track new trends," he says. "As a result, some of their efforts will be forcused on trying to neutralize advantages gained from technology, rather than in creating special programs of their own."

One manager not worried about this last scenario is Dean LeBaron, president of Batterymarch Financial Management in Boston. "My idea of the future money management organization is a few senior people and one big machine," he remarks. He has built his company on using technology to hold down personnel costs and improve performance through intelligence obtined in part by machines. McFarlan cites LeBaron as a manager who uses technology to increase profits. "His 'contrarian' approach has beaten the market 13 of the last 17 years and stayed even in two others. He feels that technology will help him stay ahead and reduce costs along the way."

McFarlan doesn't limit his impact assessment to investment managers, he cites the case of a regional bank which installed computers for its 20-odd brokerage/investment officers and saw significant gains in productivity and performance through the use of database management systems.

The criteria McFarlan states that are important in evaluating database products are: Their ability to handle information which is of disaggregate complexity (i.e. split up into smaller units); capable of distributing more data onto the desks of managers and workers alike; and able to break apart the notion of a central depository for data. Additionally, capacity must be available to handle the volume of data, either alone or through a shared data management environment.

Evolutionary Products

Many industry observers maintain that database management systems are now on the brink of their greatest growth cycle. Steven R. Magidson, DayFlo's vice president, marketing, is one of those people. An industry veteran, he asserts that "product offerings are increasingly multi-task and evolutionary.

"We're credited with the establishment of the 'text' database designation within the PC industry as well as the 'information tracking' software label," he relates. "We're seeing a greater and greater need for combining numerical and textual data."

Like others, Magidson states that the industry is moving towards more immediately usable, less programmer-developed applications. "Our customers want to take the package out of the box and use it. They don't want to wait for the experts to interpret their needs." Bill Hutchinson of Toronto-based Portfolio Money Ware Inc. says his product is designed "from the ground up to go directly to the end user.

"We see the niche we serve as closer to the applications and products, like our Portfolio Plus, that can apply to a full range of functions not served by other software products," he states.

Serving the technology needs of the financial services community requires nimble marketing and development management. There is a high degree of mortality associated with the industry. A list of 63 names gathered from published sources from 1984 through 1987 shows that less than 30 of those DBMS products and/or companies still survive in one form or another. Part of the marketing equaition for survivability of companies is the constantly declining price/performance equation.

Costs for software surveyed for this article ranged from $50 to $75,000 and up. At the same time, newer products coming to market this year offer improved features at often reduced prices. For example, Lotus is touting its new Lotus/DBMS product while at the same offering a high performance SQL (Standard Query Language) server. The prices for these

products are expected to be below competing products.

Technology Derby

Not to be overlooked in this technology derby is dBase from AshtonTate, based in Torrance, Calif. Perhaps the most well-known of the DBMS products, the newest version is dBase IV, which strives to answer user demands for a more flexibile file management system, while giving greater value for the price.

Some DBMS products have evolved down to the personal computer-level from large mainframe programs. Others have been developed directly in the personal computer environment, taking advantage of internal operating systems.

Products in the former category are Oracle, from Oracle Corp., Belmont, Calif.,; Informix, from Informix Software Inc., Menlo Park, Calif.; and PC/Focus, Information Builders Inc. New York. Genifer, Bytel Corp., Berkeley, Califf.; Paradox, recently purchased by Borland International Inc., Scotts Valley, Calif.; PowerBase, by Compuware Corp., Farmington Hills, Mich., are products conceived and implemented on personal computers. Many are now actively engaged in porting to the PS/2 environment and others are gearing up for the next generation of technology. Not surprisingly, many seek to portray themselves as providing the user with good value and being complete and operable directly from the box.

One enthusiastic user is Tony Sacchitelli, assistant vice president at the American Stock Exchange. Sacchitelli uses Oracle because he feels the product allows him to prototype applications. "We took some SIAC files, loaded them on a VAXcluster 8350 and 750. Then we showed the users some elementary SQL routines and let them test the search methodology. It took us a dy to load the files, another day to test, and we were on our way to a finished product by the third day," he relates.

Sacchitelli's enthusism reflects that of many users who have spent previous years learning how to utilize the power of products like Oracle. Others are equally positive about the expected products being offered by vendors this year. The early feedback from Lotus' beta sites for Lotus/DBMS are strongly favorable. Initial tests reportedly show a more pragmatic, organic product with great flexibility.

Rick Williams is a senior systems consultant for AVCO Financial Services in Irvine, Calif. He says he has slightly 'mixed feelings' about Versa Form from Applied Software, Los Gatos, Calif. "Our software is used by clerical people and it is very good for them because it's easy to use," he says. "It's functionally rich and its ability to handle data, particularly financial data, is very important to us. However, sometimes I feel that there is a real need for improvement in the system-to-disk area," he points out.

Paul Marentette is an assistant vice president responsible for personal computer support at Equitable Capital Management, a subsidiary of Equitable Life Assurance in New York. He has used Powerbase to track "open commitment data" for private placements from "a twinkle in the mind" of an associate through to final signoff.

Marentette, says he has used PowerBase to track "very complex multiple files with lots of links, tables, lookups and field validation functions. The applications change with the industry and now we're going to a Hewlett-Packard minicomputer to unify everything involved in our private placement area," he comments.

Marentette says he was using Powerbase for other investment applications, including equities rating, contact addresses, and even a client marketing database.

He relates that, "products with multiple fields and easy transposition of data, but which can be implemented without the services of a programmer would be preferable in the future."

Some firms contend that clients will always need programmers. However, in the future these consultants will be able to deliver a lot more and give users greater freedom thanks to the new products in the offing.

Bob Martin is director of product development for Ontologic, Billerica, Mass. His firm is in the proces of releasing a Vbase, **object-oriented** databse. According to Martin, his product "uses a whole new branch of DBMS technology which integrates text, geometry and

graphics while providing high level system modelling."

Kevin Howe has a different approach. The president of BAC Software Inc. in Dallas is releasing a product called Hold Everything. Howe says he believes his company's product answers a need for people who want to "throw a lot of information into a database and then retrieve it by word or numrical association."

Moreover, Howe is going a step further. His company will use its products to market databases of information to clients in specific industries. "The product will be there and so will the information to make it more useful," he says. "One big niche will be supplying dat and a free-form product to enable a user to access that data the way he or she wants."

Recently Unisys, headquartered in Blue Bell, Pa., has been using a headline, "Businesspeople waste the equivalent of one day a week looking for information."

While consultant Kenneth Lehrer didn't necessarily waste time looking for information, it was simply that the amount of information, as countless other businesspeople encounter, had become unmanageable. A couple of years ago, when his financial and real estate consulting business in Houston was limited to creating institutional presentations on construction projects, and before it expanded to where he chose database software to keep track of records, regulations, and anlytical information, his was a cut-and-paste job.

At the time he was doing consulting projects throughout the country and was often having to manually bring in information from one financial package to another, print it on one printer, and then paste it into the report after doing the verbal part in a text management system.

But those days are gone. Lehrer says he's happy they've cut out his night-before-presentation paste pot parties. "It was hard on the floors and its getting hard on me as I get older," he says with a laugh.

CAPTIONS: Database management software. (table)

SPECIAL FEATURES: illustration; photograph; table
DESCRIPTORS: DBMS; Functional Capabilities; Applications; Financial Services; Information Resources Management; Financial Software; Banking; Software Selection; Market Analysis
SIC CODES: 7372 Prepackaged software; 6000 DEPOSITORY INSTITUTIONS
FILE SEGMENT: CD File 275

4/9/16 (Item 1 from file: 16)
DIALOG(R)File 16:Gale Group PROMT(R)

05260913 Supplier Number: 48016523 (THIS IS THE FULLTEXT)
Financial CAD's Plug and Play Platform
Webb, Andy
Wall Street & Technology, p076
Oct, 1997
ISSN: 1060-989X
Language: English Record Type: Fulltext
Document Type: Magazine/Journal; Trade
Word Count: 2301
TEXT:
Armed with an agressive pricing strategy, Financial CAD is rolling out a single server platform aimed at helping dealing rooms to define new instruments and rapidly build integrated financial systems.

The release of FinancialCAD Corp's (formerly GlasscoPark) new Server must rank as one of the biggest shifts in territory seen to date in the financial IT market. Based in Surrey, British Columbia, the Canadian software company, known for its work with financial derivatives, has previously specialized in producing pricing models, with its current library extending to some 140 items. These were originally produced as Excel add-ins, but have more recently also become available as Visual Basic

and C++ run time libraries for those developing inhouse applications. The company has built a strong reputation in this field for both quality and extremely aggressive pricing, but is now up against a very different breed of competitor, albeit in a surprisingly small market.

The new product, called Financial CAD Server, set for release in October, is designed to allow users to instantly create and share models of any type of **financial instrument** without needing any programming knowledge. By using a single server platform, which hinges on the Unified & Trade; Data and Object Model, to manage all analytical and transaction data, the Server will also calculate valuation, risk and cash flow data for those models. The company claims that clients will be able to develop and deploy scaleable, reliable and powerful applications that will easily integrate with existing systems.

But, defining the rather nebulous "financial server" market that FinancialCAD has just joined isn't that easy. While there are plenty of companies that offer toolkits to develop new financial applications, there are relatively few that can offer an automated **object-oriented** environment suitable for the rapid development and deployment of fully integrated financial solutions. Of these firms, Infinity Financial Technology Inc., based in Mountain View, Calif., has (by IT standards) attained almost grandfather status. Apart from FinancialCAD, other players in the field include Integral Development Corp. (founded by Harpal Sandhu on leaving Infinity) and two Nordic companies -- VärdePappersData and Oy Trema.

While the attractions of a structured **object oriented** environment for rapidly developing new financial applications have been the topic of much brave talk in the past, the reality has typically been somewhat in arrears. However, there are at last signs that the technology is fast approaching a feasible and practical maturity. Deals are being done. Integral announced two major contracts this year: JP Morgan completed the development of a front-to-back office emerging markets derivatives application in six months using Integral's Universal Financial Server (UFS). And Bayerische Vereinsbank has taken delivery of UFS as the basis for an exotics system that will integrate to mid and back offices.

Meridien Research, the Boston-based technology research firm, has recently produced a major report on the "financial server" market - ("Development of Risk Management and Trading Applications using Object Technology"). Octavio Marenzi, the research director who compiled the report, feels that the timing of the technology's coming of age is certainly propitious. "The [application] development cycle has simply been too long and as new financial instruments have appeared the real bottleneck has been IT. In an attempt to circumvent this, people have developed products using manual processes such as spreadsheets, which is not a very satisfactory approach when it comes to maintaining controls or integrating with other systems."

The financial server approach isn't a universal panacea, though. Some developers, who have made speed of application development the ultimate priority, can create problems for clients further down the line. "While they may get applications out quickly they lack the quality of data structure necessary for running meaningful data queries at a later date," says Marenzi. "With their approach, a new exotic structure isn't stored in the database in a structured format using a data model, but thrown in as a large binary object. If you wish to use this data for other applications (such as risk management) it cannot be accessed using SQL."

Financial server technology is also not necessarily the most appropriate solution for all. "It can be great for larger firms that have the IT resources to get the most out of it," says David Little, president, Derivate Services Corp. "For smaller firms the training aspects can prove too costly and they tend to do better buying off the shelf solutions."

Though the off-the-shelf approach may traditionally be preferred by second and third tier players, there are signs that financial server technology could soon be trickling down to them. The development of proprietary instrument models may not be feasible for them, but the developers of financial server technology have already defined most commonly (and some uncommonly) traded instruments. For the smallest

participants, there is now no reason why this technology should not be available to them on a bureau, or even Internet, distributed basis.

Desks can Define Instruments

The fact that this type of server approach allows the trader (rather than a technologist) to create and define financial instruments can also make for more effective data maintenance. Though approaches to this issue vary slightly among the vendors, the general rule is that the creator of a **financial instrument** on the system is also responsible for its maintenance and that of any associated data. As David Glassco, CEO of FinancialCAD points out: "Who better to maintain the FX spot data than the spot FX desk?"

Nevertheless, some see problems in allowing new financial instruments to be defined in the front office. "This approach has the potential to get very messy," says Marenzi. "No matter how well these things are constructed, there's always the possibility that the trader will do something silly and produce some nonsensical **financial instrument** that will have to be disinterred from the database and rectified. It's also very difficult to optimize this kind of approach. If you have a large database with several thousand transactions an hour passing through it, you really need to get into the code in order to do that, which is something that the trader is unlikely to be able to do."

Unified Data Model

The FinancialCAD Server has had a lengthy gestation, starting life as FEO (Financial Engineering Objects) in mid 1991. ("Until we discovered that FEO meant "ugly" in Spanish," says Glassco.) "We quickly learnt that we needed a normalized data model that managed the primary data relationships," says Glassco. "We also realized that we needed normalized analytical interrelationships, so that if (for example) a 90-day LIBOR rate changed it would automatically propagate throughout the entire system across FX, bonds, commodities, equities and related derivatives like swaps and options."

The unified data model in FinancialCAD Server also means that it is extensible. New types of financial instruments can be added and integrated into the model on an ongoing basis. It also provides for unification across an enterprise, so that (for example) the FX team in an institution is using the same curves as the bond team. At the same time, the unified model dramatically reduces the total number of quotes coming into the system by automatically selecting only those quotes that are actually needed to build such things as yield curves.

Apart from the unified data model, the other thing that Glassco cites as a unique feature of FinancialCAD Server is that it has been implemented in a third generation, three tier client/server architecture. "This is something that the financial sector hasn't recognized yet," he says. "Because the Server is based on a distributed **object oriented** framework, companies wishing to scale from a single user system to several thousand and deploy financial objects on multiple machines on multiple sites can do so."

Predictably Glassco is upbeat about the prospects for this technology, but despite the ability to define any **financial instrument** in FinancialCAD Server, he remains pragmatic about the likely speed of take up in the financial industry. "It takes a lot of time and energy for companies to re-engineer their thinking and systems. However, our view of the financial market is that eventually it's going to be a Visa-like transaction pumping network, with multiple distributed components and transactions made across the Internet. We've built Server to play a part in that."

Aggressive Pricing

The pricing of FinancialCAD Server is aggressive to the point of violence, and therein could lie a problem. (A purchase price of $15,000, with an annual license fee of $5000 and $100 per month per user.) There is the risk that major investment banks will refuse to take a product seriously that is priced at this low a level, when they are more accustomed to price tags on this sort of product starting at around $250,000. However, it is clear that the company is aiming at a much broader market. Glassco is unrepentant about this. "We already service some 800 corporate treasuries

and want to price the product so that they can all afford it. We surveyed our smallest customer when fixing the price to find what they could afford. Our costs aren't so high that we have to price it at $500,000 and if we do that we immediately restrict ourselves to the top tier banks."

Microsoft, whose treasury department is going to be the first beta site for the Server (and with whom FinancialCAD has close technology links), endorses this pricing ethos. (Though Financial CAD Server will run on any ODBC database, the foundations of the product are pure Microsoft -- DCOM, NT, SQL and Transaction Servers) "You want the exposure and the continual usage of the product," says David Gumpert-Hersh, Microsoft product and technology manager worldwide capital markets, in Redmond. "FinancialCAD wants the Server anywhere there's a machine that can use it -- even in a three-person operation. They are aiming at higher volumes at a lower price with the broadest possible user base."

Gumpert-Hersh is also upbeat about the prospects for the FinancialCAD Server. "Derivatives are getting more and more intertwined with the everyday operations of treasuries. As a result corporate customers are becoming more sophisticated and making greater demands on their banks. Based on that, you're going to have a lot more value-at-risk calculations demanded from directors and board members. The real value of this product is the new architecture, which drives the product to a much higher level. If it can deliver what FinancialCAD already delivers in an add-in form it should be a real winner."

Though FinancialCAD Server has the ability to service the most sophisticated users and their extensive customization requirements, it will also be available in a format that will allow smaller clients to get up and running rapidly. The product will initially ship on pre-built servers -- typically high-end Pentiums with 64MB of RAM. These will have all the FinancialCAD software (and the supporting Microsoft products, which includes SQL, NT and Transaction Servers) fully configured and tested. All clients will have to do is unpack the crate and plug the machine into their network. They will also have the option of having FinancialCAD set the machines and software up to their specific requirements.

Other players in the field have not been inactive either. Infinity launched their FinEx product at the SIA Conference in June. FinEx, when used in conjunction with Infinity's integrated trading and risk management system, afford traders and other users the same opportunity to define new financial instruments in a spreadsheet automatically in a broadly similar fashion to FinancialCAD and Integral.

Infinity's senior product marketing manager Richard Walker is optimistic not only about FinEx but also about the validity of this approach in general. "Platform-based development has been gaining increasingly broader market acceptance. Infinity customers have demonstrated success with the Infinity Platform for some time. Now other participants are hoping to duplicate the platform approach, and have accepted that this technology is more substantial than a flash in the pan. Infinity's approach differs from Integral and FinancialCAD in that our FinEx offering is wholly integrated with the Infinity integrated trading and risk management system, which includes the Infinity Data Model and Fin++ Class Library. "

Like Glassco, Walker sees the technology developing a broader user base. "We've been very successful in selling the Infinity Platform into organizations with staff capable of developing C++ applications," he says. "However, the majority of end users of those applications have a different skill set -- they're not system developers. What they are good at is using spreadsheet applications to model the financial behavior of the instruments they handle. By having access to the underlying analytics of Fin++ via an Excel interface with which they are comfortable, these end users are empowered to develop valuable financial tools very quickly."

Integral's customers to date have used its Universal Finance Server to replace their existing financial technology infrastructure in building derivatives trading and treasury-wide risk management systems. Unlike Infinity and FinancialCAD, Raj Patel, managing director of Integral Europe, is skeptical about the "one vendor supports all financial products" approach. "It all depends on how you chose to define 'support'. If support

includes workflow, security, audit, and full front to back office processing, I don't think anyone can claim to support all financial instruments. Also, the claim that a single data model can support every type of **financial instrument** from a trading and back office perspective and at the same time act as a risk management warehouse is simply not feasible," says Patel. "In the end, transaction data models and risk management data models should be designed to support their respective requirements," he says. Nevertheless, he does believe "the next generation of technology, namely the financial server, will eventually span the entire range of financial instruments primarily because the vendor merely provides a framework and the user has the ability to fill in the workflow pieces. Net result -- a complete solution, to the user's specs, in record time."

PUBLISHER NAME: Miller Freeman, Inc.
COMPANY NAMES: *FinancialCAD Corp.
EVENT NAMES: *336 (Product introduction)
GEOGRAPHIC NAMES: *1USA (United States)
PRODUCT NAMES: *7372620 (Network Software)
INDUSTRY NAMES: BANK (Banking, Finance and Accounting); BUSN (Any type of business); CMPT (Computers and Office Automation)
NAICS CODES: 51121 (Software Publishers)
SPECIAL FEATURES: COMPANY

4/9/17 (Item 2 from file: 16)
DIALOG(R)File 16:Gale Group PROMT(R)

04655059 Supplier Number: 46847991 (THIS IS THE FULLTEXT)
Enterprising Solutions
LatinFinance, p73
Nov, 1996
ISSN: 1048-535X
Language: English Record Type: Fulltext
Document Type: Magazine/Journal; Trade
Word Count: 1340

TEXT:

Bank-wide risk management: Practical implementation

Hagay Shefi

Recent high profile catastrophes in derivatives trading have increased awareness among banks and financial institutions of the need for more effective risk management measures, and prompted fresh calls from regulatory authorities for tighter controls. Studies by regulators and academics have analyzed the problems and recommended solutions. In the main, these have pinpointed the need for greater management understanding of the risks involved and better use of technology. Attention has been focused on the substantial investment in research and development from financial software vendors which has established new methods and techniques for monitoring and managing risk.

In essence, the findings underscore the G-30 report which highlighted different aspects of risk management practices and emphasized the role of systems and technology. It stated that firms, both dealers and end users, should establish management information systems sophisticated enough to measure, manage and report the risks of derivatives in a timely and precise manner.

The Many Dimensions of Risk

In the past two years, the concept of enterprise-wide risk management (ERM) has evolved as banks and financial institutions have realized that risk can be multi-dimensional and should include a wide range of definitions.

Most methodologies appropriate to a bank-wide risk management originated in derivatives trading activities. While this is interesting and important, it should be noted that risk management is not a formula. Risk

Metrics or any other V[redacted] (value-at-risk) methodology is [redacted]ust one potentially valid too[redacted]

An acceptable definition of bank-wide risk management may be: 'The systems and procedures designed to deal with the multiple types of risk, such as market, credit, liquidity, operational and legal, that arise from dealing in different asset classes such as currencies, interest rates, equities, and commodities across different time zones.'

As risk may arise from many dimensions, the practicality of risk management requires participants to address the following aspects of risk:

Market Risk: uncertainty of future earnings resulting from changes in market conditions surrounding equities, interest rates, foreign exchange and commodity values.

Credit Risk: potential loss due to the inability of a counterparty to meet its obligations.

Liquidity Risk: market impact of buying and selling a **financial instrument**.

Operational Risk: errors that can be made in processing and settling transactions.

Regulatory: changes to, or enactment of, new legislation.

Human Factor: problems ranging from keyboard errors to major fraud.

With so many potential risks there is no logical denial of the need for effective risk management on a global scale. Moreover, today's specifications for any risk management system should take into consideration that present and future requirements lead to more, not fewer, systems and that the importance of integration will remain paramount. Increased regulation, collateralized derivatives clearing houses and standardization of methods such as value-at-risk offer increasing local and global benefits.

The existence of legacy systems and the importance of their integration into a contemporary ERM systems strategy is also vital. Current G-30 recommendations and mooted Bank for International Settlements, Federal and other regulators' requirements must be accommodated in order to provide the principal goal of achieving consistent, consolidated risk management across all instruments, books and locations.

In order to effectively address those aspects, an ERM system must calculate the following risk characteristics: greeks (delta, gamma, etc.); VaR; storage of historical information; trading and credit limits; simulation/stress testing; back testing.

It must also track all information affecting risk: market prices; trades (locally and globally).

Addressing today's requirements and those anticipated in the future in an ERM calls for solid building blocks. These should cover a sound technology based on industry standards which will continue to evolve, along with crucial functionality such as risk archives, VaR, simulation, source of change, limits and global trade consolidation.

The risk archive stores primary and derived data for historical analysis. Data is stored at user defined frequencies and used for VaR and source of change analysis (e.g. for computing the variance and covariance matrices) and can be used for proprietary analysis.

Value-at-risk measures the maximum loss in market value of a given position that can be incurred until the market risk exposure is neutralized, with a given confidence interval, assuming markets continue to behave as they have in the past. VaR uses historical information stored in the risk archives to determine empirical correlations both intra- and inter-markets at user specified 'confidence levels' and user specified time horizons for position unwind. Simulation provides advanced techniques like scenario analysis, worst/best case analysis as well as forward simulation, Monte Carlo, historical simulation and stress testing.

The source of change analysis (also known as 'what was?' analysis) provides answers to the crucial question: What are the factors that contributed to gain/loss in a portfolio at any given period? More specifically, the technique provides analysis of the source of P&L movements over an entire portfolio, comparison of actual contributions from different risk sources (e.g. delta, time decay, second order changes, etc.) to expected contributions from theoretical projections. Ideally, this

should also include a facility to 'drill down' through various data levels to reveal the actual source of change.

Limits capability is important for both market and credit risk and should be monitored in real time against any risk measurement.

For multi-site operations, global trade consolidation is crucial for assessing the risk from any location or system. Thus, the ERM system must ensure that it tracks all global trades. While it may be required to track partial global risk in a hierarchical structure, it also has to track risk as it changes and that 'batch' risk isn't typically enough and real-time approach is more practical in today's dynamic market. Global trade consolidation also means that tracking of risk in multiple disparate locations includes multiple applications and multiple platforms and protocols as well as multiple structures.

Bringing Everything Together

Implementation of any effective ERM system will almost certainly involve a complicated configuration of numerous legacy systems and procedures spanning front, middle, back office and accounting practices. Clearly, consolidating trades and positions from many different applications raises serious issues regarding the quality of the data and the actual ability to technically transfer the data from legacy systems to the central data warehouse for risk management purposes.

Further analysis of the effective ERM should, therefore, cover the following aspects:

Definitions: the details on how risk is measured and tracked by the bank.

Transport: an interactive layer which takes data from source systems and makes it available for analysis.

Normalization: a methodology for making risk data similar so that it can be aggregated and analyzed

Data Warehouse: input and output of risk analysis.

Analytics Package: tools for analyzing the different risk types.

Tracking: tools for comparing business activities to management objectives and other guidelines (e.g. limits).

Reporting: tools for communicating the results of analysis back to decision makers.

If the ERM system is to achieve its key goals for satisfying managers and the regulators, it needs to be based on enduring, open systems technology with client/server architecture, an ultra-friendly graphical user interface and controlled access to structures and models.

The technical and financial requirements for effective risk management can be met. Windows NT technology, along with SQL databases and efficient use of object linking and embedding and **object oriented** technology are key to the success of the implementation. The combination delivers efficient infrastructure for global trade consolidation, data modeling, interfaces to legacy systems and powerful analytical analysis.

A practical approach for risk management will focus on a 'brain' component that will interface with trading and back office systems as well as market data providers. The 'brain' (also known as a risk executive) knows how to process events (e.g. execution of new trades, amendment of existing trade, change in market rates/volatility etc.) and how to transport the data to the risk warehouse.

Once a comprehensive risk warehouse is established, analytical packages (risk workstation) can be applied for VaR, back testing, simulation, credit risk analysis and other advance techniques for effective risk management. LF

Hagay Shefi is managing director for North and South America at SunGard Capital Markets

PUBLISHER NAME: Latin American Financial Publications, Inc.
EVENT NAMES: *250 (Financial management)
GEOGRAPHIC NAMES: *0LATI (Latin America)
PRODUCT NAMES: *6020000 (Commercial Banks)
INDUSTRY NAMES: BANK (Banking, Finance and Accounting); BUSN (Any type of business); INTL (Business, International)

NAICS CODES: 52211 (Commercial Banking)
SPECIAL FEATURES: IND[redacted]RY

4/9/18 (Item 3 from file: 16)
DIALOG(R)File 16:Gale Group PROMT(R)

04219281 Supplier Number: 46173207 (THIS IS THE FULLTEXT)
Objects Take Off
InformationWeek, p14
Feb 26, 1996
ISSN: 8750-6874
Language: English Record Type: Fulltext
Document Type: Magazine/Journal; Tabloid; General Trade
Word Count: 1188
TEXT:
In China, 1996 is the year of the rat. But in the world of enterprise applications, 1996 is shaping up as the Year of the Object. In what eventually could turn into the largest implementation yet in the nascent market for packaged **object-oriented** applications, Marcam Corp. expects to deliver a version of its Protean production and inventory software on March 5 to PepsiCo Inc.'s Pepsi-Cola North America division.

Pepsi is one of several big companies seeking flexibility and ease of use by buying **object-oriented** enterprise applications. "Everybody is suddenly discovering how hot objects are going to be this year," says Bruce Richardson, VP for research at Advanced Manufacturing Research, a technology research firm in Boston.

For some time, in-house corporate developers, especially in industries such as financial services, have used objects--reusable chunks of software code--to save time in building custom applications. While that practice continues to grow, what's different about the latest trend is that companies are buying ready-made **object-oriented** applications, which require little or no development expertise. The payoff for companies comes not in developing the original software, but in modifying the application to adapt to a changing business without having to touch the underlying code. The desired result: a nimbler company.

"Adaptability is the key," says Ray Sasso, corporate chief information officer at J.R. Simplot Co., a $2.8 billion food processor in Boise, Idaho.

Information systems managers should expect to see an increasing number of software vendors offering object-based applications. "Objects are the future of the business because they require very little custom programming," says Doug Magill, a consultant and former CIO at Nestl -Stouffer Co. in Solon, Ohio.

In addition to Marcam, a number of vendors, most notably System Software Associates in Chicago and Systems & Computer Technology Corp. in Malvern, Pa., already have customers using their enterprise-level **object-oriented** packaged software. Dun & Bradstreet Software Inc. has an **object-oriented** workflow capability in its enterprise application suite. Other vendors, including QAD Inc. in Carpinteria, Calif., and Sherpa Corp. in San Jose, Calif., will soon release object-based applications.

If all goes well with a $3 million project to test Protean at one of Pepsi's bottling plants, the company intends to install the software to link all its North American bottling and warehousing activities--65 bottling plants and 240 distribution centers--in a common system for managing production and inventory of the division, which has $6 billion a year in sales. The initial project, scheduled to be up and running by June 1, will include some 200 users with Microsoft Windows PCs. That would expand to include thousands of users throughout the company's supply and distribution network.

The flexibility offered by **object-oriented** software was apparently a key factor in Pepsi's decision to install Protean. "Its business changes rapidly, and the company felt that the adaptability and flexibility inherent with our object architecture would make it easier for

them to reflect these changes in the system," says Mark Arsenault, an account manager at Marcam in Newton, Mass.

"We see what we are doing with Protean as having a potential for achieving competitive advantage," says Ken Gerhardt, director of application development at Pepsi-Cola North America.

To win the Pepsi contract, Marcam beat out Dun & Bradstreet, which already had a foothold because Pepsi uses D&B's mainframe-based financial software. For Pepsi, installing Protean is a logical extension of the company's substantial in-house **object-oriented** application development effort using the PowerBuilder development package from Sybase Inc.'s Powersoft unit, according to Marcam, which says some of Pepsi's internally developed objects can be integrated with Protean.

Pepsi isn't alone in its desire to use objects to act quickly. Marcam, which has sold Protean since late 1994, recently signed deals with such major companies as 3M Co. and Sun Chemical Corp., as well as with Simplot.

With any fast-changing business, no application will last long. "In that context, you can't have an investment in old, inflexible applications that are difficult to maintain," says Steve McClure, director of object tools at International Data Corp., a Framingham, Mass., market research firm.

For example, in financial services, where **object-oriented** development is common, "every day someone's inventing a new **financial instrument**," McClure says. "Those companies can't wait two years to respond."

But fast reaction times aren't just for financial firms any more. Simplot, the Idaho food processor, began using Protean at one plant last fall and at a second plant in January. "The object orientation provides us with so much customizability," Sasso says. "It makes it easy to adapt the system to business changes very readily."

Sasso also says it was easy to train workers to use the system to record shipments, update inventory, schedule production, and track orders. Roughly 80% of the workers had never touched a computer before. Simplot figures it will spend $15 million to $25 million to roll out the software to its entire 38-plant operation over the next four years.

Ramsey Sias Co., a Brecksville, Ohio, maker of fruit fillings for companies such as Dannon and Haagen Dazs, began using Systems & Computer Technology's **object-oriented** Adage software in January to manage purchasing and accounts payable. "The company's users can see their business process expressed as objects on the screen, and they are able to grasp the different steps involved pretty quickly," says Magill, a consultant for Ramsey's object software project.

Despite the burst of object activity, observers say technology users have been slow to embrace the software packages because the technology is so young and relatively untested on a scale the size of Pepsi's plans. Others may be hesitant because they perceive no immediate payoff. "It's still at the leap-of-faith stage," says Richardson, the manufacturing technology analyst.

Maybe, but a growing number of big companies are taking that leap.

Related Article: Objects: A New Twist

Pepsi's selection of an **object-oriented** application is a new twist on the "objects are good for you" story. We've heard object technology touted for years as a tool for making developers nimble in building applications. But now we're finding that those same technology characteristics--encapsulation, inheritance, and polymorphism--can deliver benefits to users of object-based applications as well.

Off-the-shelf software built using object technology can offer flexibility to a business. Object-based applications are easily changed because they're built using reusable, modular components. A good application design passes that adaptability on to the user--letting an organization assemble components to match applications to its business model.

Inheritance lets the information systems group alter a base class, such as monthly statements, to reflect, for example, a new discount policy. Because the functions of that object are hidden-encapsulated--users don't need to know how the object works, only how to use a high-level scripting

language to change its behavior.

Polymorphism, in which different objects can plug to the same interface, allows the output of a report to be sent to a local printer, faxed to a branch office, or E-mailed to an executive on the road without any changes to the source code.

This level of customization could be provided using traditional development methods, but it would take longer to build-and in a competitive market, time is of the essence.

Julie Anderson is technical director of InformationWeek's OpenLabs.

PUBLISHER NAME: CMP Publications, Inc.
COMPANY NAMES: *Dun and Bradstreet Software Services Inc.; Marcam Corp.; Pepsi-Cola Co.; qad.inc. (Carpinteria, Californ; Systems and Computer Technology Corp.; Systems Software Associates
EVENT NAMES: *330 (Product information); 610 (Contracts & orders received); 430 (Capital expenditures); 600 (Market information - general)
GEOGRAPHIC NAMES: *1USA (United States)
PRODUCT NAMES: *7372510 (Software Development Tools); 2080010 (Nonalcoholic Beverages)
INDUSTRY NAMES: BUSN (Any type of business); CMPT (Computers and Office Automation); TELC (Telecommunications)
NAICS CODES: 51121 (Software Publishers); 31211 (Soft Drink and Ice Manufacturing)
TICKER SYMBOLS: MCAM; SCTC
SPECIAL FEATURES: COMPANY

4/9/19 (Item 4 from file: 16)
DIALOG(R)File 16:Gale Group PROMT(R)

04199857 Supplier Number: 46141451 (THIS IS THE FULLTEXT)
Focus On Business Objects
InformationWeek, p51
Feb 12, 1996
ISSN: 8750-6874
Language: English Record Type: Fulltext
Document Type: Magazine/Journal; Tabloid; General Trade
Word Count: 1177
TEXT:
The computer industry's leading object-software coalition is beginning to tackle a job it considers the next logical step in **object-oriented** computing: defining business objects that can be used to create applications for intercompany commerce and communications.

Until now, the 600-member Object Management Group (OMG) in Framingham, Mass., has focused on defining specifications for the "plumbing" of **object-oriented** computing. But OMG members say the real payback from **object-oriented** development will come when reusable software components become an integral part of each business process.

"If you think about it, our view of what a package is shouldn't be that different from an airline's or trucking company's view," says Frank Ginett, senior technical fellow with Federal Express Corp.'s Information Resource Management group in Memphis, Tenn. "There are processes we use that are similar."

The OMG has set out to define business objects on two fronts. One working group is trying to define common business objects-software components that can be applied across industries.

Other working groups are applying the same principle to vertical markets, so developers can work with objects that have been defined for use within specific sectors, such as finance, manufacturing, and telecommunications.

OMG president Chris Stone says his organization should have its first business objects defined by year's end. Eventually, he says, there could be

hundreds of business objects.

"It's a simple concept," says Cory Casanave, chairman of the OMG's business object domain task force and co-president of Data Access Corp., a systems integrator in Miami. "You model the business as objects, and then implement those objects directly in your system." For Casanave, business objects are "visible components in an information system," such as a customer, product, or sales form.

Goal: Interoperability

Andersen Consulting, Fort , IBM, SAP, VMark, and a growing number of other software vendors sell products addressing similar concepts. The OMG is soliciting these vendors to submit commercial objects for consideration as OMG standards. "There are some early adopters and early products," says Casanave. "We're hoping to bring a level of consistency and standardization so that business objects developed in any environment will be interoperable."

Several user organizations have started creating their own business objects. As part of a three-year-old customer-service project that uses Next Computer Inc.'s NextStep object environment, AT&T Wireless Services in Kirkland, Wash., has created objects to represent telephones, cell switches, services, and accounts. "The wireless industry is moving so fast we really didn't have the luxury to wait," says Hide Horiuchi, the company's technical architecture manager.

The Cushing Group Inc. in Nashua, N.H., has created dozens of business objects as part of a customer-profile application it's building for Wells Fargo Bank in San Francisco. The objects allow database in- tegration, the reuse of software code, and immediate access to customer data, says Erik Townsend, president of the Cushing Group. "We have created standards within Wells Fargo that provide object interfaces for things like accounts and customers," he says. "The OMG is taking that concept and extending it across an entire industry."

The Cushing Group used several object technologies to create its Wells Fargo objects, including Digital Equipment's ObjectBroker, Expersoft's Powerbroker, Iona's Orbix, Microsoft's OLE, and IBM's System Object Model. With the exception of OLE, all comply with OMG's Common Object Request Broker Architecture (Corba), a comprehensive set of standard programming interfaces that permits different vendors' object request brokers to work together. These brokers allow objects written in different programming languages to communicate.

Among the companies participating in the OMG business object project are BellSouth, Boeing, Federal Express, GTE, and Texas Instruments. The business objects task force has labored nearly three years to gauge market interest in business objects and sketch preliminary definitions.

That work should bear fruit within the next few months: The group recently issued its first request for proposals, calling for users and independent software vendors to submit object specifications that can be used as standards.

"The goal is a set of interfaces for application components so that independently conceived and developed components can interface in an information system," says Casanave.

Standards By Sector

A related effort to create objects for use between companies in the same industry got a boost in December when OMG reorganized to better address the requirements of vertical markets. The organization is forming working groups to define standards for industry sectors, or domains, including finance, health care, interactive multimedia, manufacturing, and telecommunications. Other industry-specific working groups are expected.

"Companies like mine buy a lot of application software," says Tom Hein, manager of technology integration with heavy equipment manufacturer Deere & Co. in Moline, Ill. "This will simplify our jobs as users by giving us more Corba-conformant application software, helping keep the number of interfaces small, and ensuring reuse."

User interest in most of the vertical groups is high. Organizations participating in OMG's financial domain task force, for instance, include Barclays Bank, Chase Manhattan, J.P. Morgan, Merrill Lynch, NatWest, and Wells Fargo, as well as the Federal National Mortgage Association, the

National Security Agency, and Japan's Ministry of Finance. "We've been around the world talking to users about this," says James Hassell, who chairs the financial task force. "They need it big-time." Hassell owns Stanford Software, an object software vendor in Stockport, England.

The financial group presented a first draft of its work at an OMG meeting in January. Objects are divided into four categories-business support, customer support, decision support, and financial products. Each includes facilities that can be adapted to Corba (see chart, p. 51).

The approach makes sense for Stephen Siegel, managing director of Fusion Systems Group Inc., a New York consulting firm specializing in emerging technologies. Fusion Systems' clients include Wall Street firms in the midst of deploying distributed, object-based applications. "[We] are constantly reinventing the wheel," says Siegel. "If we're going to realize one of the advertised benefits of objects, namely reuse, [domain objects are] absolutely required."

Siegel says domain objects will cut the time and cost of development because custom programming can then be supplemented with off-the-shelf objectware. "It's a classic buy-or-build situation," he adds.

Define And Integrate

The OMG's telecom group is focusing its efforts on defining object interfaces for carriers' network management systems. Equipment suppliers and carriers involved in that effort include Alcatel, AT&T, BT, L.M. Ericsson, and MCI. "This will enable operator companies to integrate multivendor solutions much more easily," says Dave Stringer, group co-chair and global standards manager with BRN Europe Ltd., a British research and development subsidiary of Canadian telecom equipment maker Northern Telecom.

But experts point out that there are potential stumbling blocks. Cushing Group's Townsend says leading-edge financial organizations may be reluctant to share experiences creating and managing business objects. "Our customers tend to be protective of things," says Townsend. "They don't want the other guys to have it."

Proposed Categories Of Financial Objects

Business support

* Accounting
* Credit and funds transfer
* Settlement
* Statutory compliance

Customer support

* Alternate distribution
* Customer descriptions
* Online banking

Decision support

* Channel management
* Contract management
* Relationship management

Financial products

* Deal-making
* **Financial instrument** trades
* Insurance
* Portfolio management

* Product de[redacted]ns

* Real-time pricing feeds
Data: Object Management Group

PUBLISHER NAME: CMP Publications, Inc.
EVENT NAMES: *330 (Product information)
GEOGRAPHIC NAMES: *1USA (United States)
PRODUCT NAMES: *7372510 (Software Development Tools)
INDUSTRY NAMES: BUSN (Any type of business); CMPT (Computers and Office Automation); TELC (Telecommunications)
NAICS CODES: 51121 (Software Publishers)

4/9/20 (Item 5 from file: 16)
DIALOG(R)File 16:Gale Group PROMT(R)

03975137 Supplier Number: 45767505 (THIS IS THE FULLTEXT)
The Open View
InformationWeek, p45
Sept 4, 1995
ISSN: 8750-6874
Language: English Record Type: Fulltext
Document Type: Magazine/Journal; Tabloid; General Trade
Word Count: 1623
TEXT:
Unix transaction monitors help businesses synchronize transactions, implement three-tier client-server architectures, and boost throughput. They could even displace mainframe monitors.

How does a bank ensure that a customer's transfer order actually credits the right account while debiting another? How does an engineering firm use a groupware application to pass off a computer-assisted design for an engine bolt to a supervisor for examination and approval before that design is shipped off to a factory? u The answer is the transaction monitor, also known as a transaction manager. Once found only on mainframes, Unix transaction monitors synchronize transactions between heterogeneous databases, implement three-tiered client-server architectures, boost the throughput of online transaction processing, and increase the number of users that a system can support.

Some users and analysts deride Unix transaction monitors as too complex and immature for widespread use. But the technology-really a form of middleware-plays an increasingly important role in corporate client-server environments.

Transaction monitors help guarantee that data-processing transactions succeed. Or, if the process fails, they make sure the failure won't damage the integrity of the data. Monitors also allow load balancing, or the spreading of applications among different machines for maximum efficiency. They also offer various remote procedure calls that link the layers in a three-tiered client-server set-up, helping a business choose the right tier for the right piece of the system.

Helping Hands

There are several reasons for the growing popularity of Unix transaction monitors. The technology tends to be less expensive than their mainframe transaction-oriented database counterparts, such as IBM CICS, IBM IMS, and Computer Associates CA-IDMS. The Unix version also requires fewer administrators, is easier to program, and tends to use cheaper hardware.

"We're getting performance that's just as good as we're used to on the mainframe," says Jim Holtman, VP of system architecture at Cincinnati Bell Information Systems Inc. (CBIS) in Cincinnati. But Holtman and other users have words of caution. They say tools to manage Unix transaction monitors are uncommon or inefficient, can't handle many users, and don't perform batch functions as well as mainframe monitors. They're even worried about

the maturity of related functions in the Unix world such as backup and recovery.

That's why CBIS, a subsidiary of Cincinnati Bell Inc., uses both Unix and legacy transaction monitors. Its Unix system is based on the market-leading transaction monitor, Tuxedo, from Novell in Provo, Utah. But the majority of processing at CBIS is still based on the IBM CICS database and transaction monitoring system.

Over the past five years, CBIS has processed more of its bills and customer service calls with an internally developed system called Precedence 2000, built around Tuxedo. With Precedence, CBIS performs billing and online customer services for cellular phone companies. Unlike the mainframe systems at CBIS, Precedence 2000 runs in a distributed fashion, spreading transactions over multiple servers to increase computing efficiency. That's a key difference compared to mainframe monitors. Given the nature of Unix systems, the management of distributed processing is vital.

Precedence 2000 helps 80 service representatives, using PCs running Microsoft Windows, respond to nearly 200,000 customer calls daily. Holtman says the Tuxedo-based system works so well that all new customers are put into it, leaving only long-standing customers in CICS systems.The company prefers the C and C++ development languages, which work well with Tuxedo for extensible, **object-oriented** development. CBIS likes Tuxedo so much that eventually it may try to move all its operations off the mainframe. But Holtman acknowledges there are drawbacks. "The critics have some good points to make," he says. "In some respects, Unix transaction managers aren't as capable as mainframes. If someone would invent a way to accurately translate Cobol automatically into C, it would sure help us out."

Though Precedence processes 200,000 transactions each day, the mainframe can handle 10 times that volume. CBIS believes the mainframe far outstrips any Unix system at inputting huge volumes of data on tape, performing backup and recovery, and executing major batch jobs and sorts.

Leap Of Faith

More businesses are experimenting-or like CBIS, running strategic applications-with Unix transaction monitors. Sales of the monitors will hit $537 million by 1998, a fivefold increase over 1994's sales of $109 million, according to projections from the Standish Group International, a consulting firm in Dennis, Mass. (By contrast, the total market for non-Unix transaction monitors was more than nine times greater last year at nearly $1 billion.)

One user, GE Capital Mortgage Corp., a seller of mortgage insurance in Raleigh, N.C., uses the technology to help link 200 users in 26 branch offices around the country to a Sybase database in Tennessee. The goal is to record all sales in real time. The company uses Encina, a transaction monitor from Transarc in Pittsburgh, to ensure the integrity of the transactions. The information is stored on a mainframe but soon will be moved to Sybase, and possibly Oracle, under some combination of Unix and OS/2.

"We feel confident enough that we're going to use the mainframe only for storage and take away any transaction processing role," says Stan Patterson, a senior technical analyst at GE Capital. "It's particularly valuable for the heterogeneity it allows."

Ed Wehner, manager of business information services at Memc Inc., a silicon wafer producer in St. Peters, Mo., rebuts the criticism that a Unix transaction monitor can't be used effectively for batch processing. Using CICS/6000, an AIX and HP-UX version of IBM's CICS, Memc runs batch transactions of customer order, scheduling, and labeling processes even while the system is operating online. When Memc used CICS on the mainframe, the company couldn't run batch and online queries at the same time. "There's nothing we can't do better under Unix than on the mainframe," Wehner adds.

Managing Growth

Using Unix transaction monitors cuts personnel costs, too. While Wehner used six or seven people to run his mainframe operations, today the same monitoring tasks can be handled by a single analyst.

One of the reasons for the proliferation of Unix transaction monitors over the past five years is that more companies need this technology to help them manage three-tiered client-server systems. In three-tiered systems, clients (containing the interface) connect to applications (containing the program logic), which in turn interact with databases and other computing resources.

The number of three-tier systems will grow by nearly 75% between now and 1997, predicts Strategic Focus, a consulting firm in Milpitas, Calif. Three-tiered systems make up only 5% of the total number of client-server applications. By 1997, this architecture will make up nearly 20% of the total, say Strategic Focus analysts.

Essential to the successful implementation of three-tiered architectures is some way of managing the interactions among the layers. Transaction monitors help companies accomplish this, says Ivan Ruzic, Novell's director of marketing for Tuxedo. The alternative, custom-written middleware, is tedious and difficult to create.

One East Coast financial institution implemented a three-tier setup two years ago using Tuxedo. PCs running Windows access applications Pyramid servers, which in turn access data on IBM mainframes. "If we didn't have Tuxedo, there's no way we could have created this system, which is [helping handle] essential functions like cost-based accounting, **financial-instrument** reporting, and commission calculation," says a senior technical staffer at the company.

The major transaction monitors support multiple databases. That makes transaction monitors popular not only with companies that operate several database platforms, but with commercial software developers as well. Among the most outspoken is Larry Tanning, president of Tanning Technology Corp. in Denver. "Anyone saying Unix transaction managers aren't ready for prime time would sound like an idiot to the 4,000 sites where we've installed our software based on them," he says.

Tanning's clients agree. Gordon Divitt, president of Fund Serv Inc., a mutual-fund network in Toronto, says he's completely satisfied with Transaction Forwarding System, a Canada-wide, three-tier mutual-fund purchasing application that Tanning developed with two partners. "I got what I wanted," says Divitt. "Development was quick. It operates efficiently. It's stable and easy to extend."

But Tanning agrees there are still important weaknesses in the technology. Unix system- and network-administration tools lag those for mainframes.

Indeed, analysts like Rich Finkelstein, president of Performance Computing in Chicago, say Unix transaction monitors are still far too difficult to install, administer, and use as a base for writing programs. "We need simplification," he says. Roy Schulte, an analyst with Gartner Group Inc., an IT advisory firm in Stamford, Conn., says monitoring tools for Unix transaction monitors aren't up to the level of mainframe products. For some compute-intensive processes, Schulte adds, companies worry that Unix tape handling isn't good enough, and that batch processing is still not up to mainframe speeds.

Future Remedies

Vendors of Unix transaction monitors are working to remedy the limitations. Sometime next year, Tuxedo will integrate more closely with Novell Directory Services, according to Tuxedo marketing director Ruzic. That will let an administrator control Tuxedo-based applications from a single monitor, along with NetWare LANs and devices that run under Novell's Embedded Systems Technology. When Novell starts supporting a Simple Network Management Protocol agent for Tuxedo, administrators will be able to control Tuxedo using HP's Open View, CA's Unicenter, or similar products.

Top End, Tuxedo's rival from AT&T GIS, already can be managed from most major systems-administration tools, says business unit manager Randy Smerik in San Diego. Now, the company will focus on porting the product to Windows NT.

Despite shortfalls, companies are lured by the technology. "There used to be big gaps in functionality [compared with mainframe monitors]," says Mike Prince, director of MIS for Burlington Coat Factory Warehouse in Burlington, N.J. "Today, it's down to the icing on the cake instead of

missing an oven to bake the cake."
That oven is [redacted]ting fancier by the day. [redacted]

PUBLISHER NAME: CMP Publications, Inc.
EVENT NAMES: *600 (Market information - general); 330 (Product information)
GEOGRAPHIC NAMES: *1USA (United States)
PRODUCT NAMES: *3573060 (Finance & Trade Computer Systems)
INDUSTRY NAMES: BUSN (Any type of business); CMPT (Computers and Office Automation); TELC (Telecommunications)
NAICS CODES: 334111 (Electronic Computer Manufacturing)

4/9/21 (Item 6 from file: 16)
DIALOG(R)File 16:Gale Group PROMT(R)

03680112 Supplier Number: 45202362 (THIS IS THE FULLTEXT)
MANAGEMENT AND STRATEGY: FOR DERIVATIVE APP, CHEMICAL USES OBJECT DESIGN'S DATABASE
Trading Systems Technology, v8, n12, pN/A
Dec 12, 1994
ISSN: 0892-5542
Language: English Record Type: Fulltext
Document Type: Newsletter; Trade
Word Count: 830
TEXT:
Chemical Bank in New York has started development work on a proprietary derivatives trading system that uses **object-oriented**, as opposed to relational, database technology. The project represents the second time Chemical has used such an architecture -- the bank has already deployed an application for trading emerging market derivatives that relies on an object database to store persistent data. The object database used in both systems is Objectstore, developed and marketed by Burlington, Mass.-based Object Design Inc.

Separately, sources say the bank has started work on a project called Bank++, which involves building an **object-oriented** layer of global risk management software that taps into all of Chemical's other systems.

The new trading system development project is headed by John Zhang, a systems engineer at Chemical's capital markets group. "I have a team of mostly C++ programmers developing a trading system with **object-oriented** technology," he says. Zhang stresses that the system is not in production yet. He declines to specify the precise nature of the **financial instrument** involved, other than the fact that it is a proprietary derivative based on interest rates.

Not Happy

Zhang says he has previously worked on **object-oriented** systems that interface to relational databases, but wasn't happy with the maintainability of the resulting code. "It's very awkward programming," he says. "One side of the system is in an object world, the other in a relational world. You have to develop some sort of interface" -- a process that is "complex and slow."

In particular, he says the fast-moving nature of exotic derivatives trading means banks have to develop code that is easy to maintain and enhance. "We need to reduce complexity -- six months on you will have to expand (the system)." In addition, Zhang says mixing object-orientation with relational databases slows down system performance, though he stresses that this is a "secondary issue" compared to the need to build flexibility into a derivatives system.

Chemical already uses Objectstore for an emerging market derivatives system supporting six traders in New York. Zhang says this system -- developed in Smalltalk rather than C++ -- was deployed around a year ago and has proved itself in practice. This was one factor leading to Chemical

tapping Object Design's software a second time, he adds.

The bank's capital markets group uses an Informix Software Inc. relational database in other applications, says Zhang. He adds that linking Objectstore to this repository isn't an issue as yet: "We don't need interfacing in this phase (of the system's development)." However, he adds that Chemical has plans to link the two database systems towards the end of the first quarter of next year, and will probably use Object Design's as-yet-unreleased DBconnect interface tool for this purpose.

Introducing **object-oriented** databases has led to some unforeseen project management problems, says Zhang. In particular, he says developing relational systems involves "a clear division of labor -- someone does database design, someone (else) does the programming." This division is blurred when using Objectstore, he says, which "demands that programmers (also) have database skills." Zhang says Chemical has had to re-engineer its application development methodology in order to partially recreate this division and assign specific tasks to specific systems personnel.

Object databases have been proposed as a replacement for the near-universally accepted relational model for some time now. Until recently this technology has been dismissed by most players in the derivatives industry as too immature to use. In particular, Los Altos, Calif.-based Renaissance Software Inc. searched for a suitable **object-oriented** database for its Opus application suite for some time, before finally electing to go with the relational Sybase Inc. product earlier this year (TST Sept. 19).

More Sightings

However, recent months have seen more sightings of this new technology in the derivatives software industry. Toronto-based Algorithmics Inc. uses an object database for its Riskwatch global risk management system (Derivatives Engineering & Technology, February), as does Price Waterhouse L.L.P. for its Risk Toolset project (DE&T, Oct. 31). Sources say that Swiss Bank Corp. in London and Lehman Brothers Inc. are also using **object-oriented** database architecture in some parts of their trading operations.

Chemical itself has used object technology for several of its derivatives and risk management applications in the past. The bank licensed Mountain View, Calif.-based Infinity Financial Technology's Fin++ **object-oriented** class library earlier this year, for use in interest rate risk management systems (TST, March 21). It also worked with Infinity and Price Waterhouse's capital markets and treasury group to develop a global accounting and netting system for foreign exchange, dubbed Globalnet.

Sources say Chemical, in conjunction with Price Waterhouse, has put in place plans to extend Globalnet into a more wide ranging system, dubbed Bank++. Though details remain sketchy, it is believed that Bank++ is conceived as a layer of **object-oriented** software designed to aid Chemical's global risk management efforts. Several of the bank's systems, including those from Infinity, are linked to this system, sources say, to form a common pool of information accessible by traders, analysts and risk managers.

PUBLISHER NAME: Waters Information Services, Inc.
COMPANY NAMES: *Chemical Bank; Object Design Inc.
EVENT NAMES: *260 (General services); 330 (Product information)
GEOGRAPHIC NAMES: *1USA (United States)
PRODUCT NAMES: *6020000 (Commercial Banks); 7372411 (General Accounting & Financial Software); 7372420 (Database Software)
INDUSTRY NAMES: BANK (Banking, Finance and Accounting); BUSN (Any type of business); CMPT (Computers and Office Automation)
NAICS CODES: 52211 (Commercial Banking); 51121 (Software Publishers)
TICKER SYMBOLS: ODIS
SPECIAL FEATURES: INDUSTRY; COMPANY

03125449 Supplier Number: 44262614 (THIS IS THE FULLTEXT)
GRASPING OBJECTS: GET READY FOR THE UPHEAVAL
Wall Street & Technology, p24
Dec, 1993
ISSN: 1060-989X
Language: English Record Type: Fulltext
Document Type: Magazine/Journal; Trade
Word Count: 2954
TEXT:
BY CARRIE R. SMITH

Roughly 450 Wall Street systems developers packed an auditorium at Metropolitan Life on Oct. 5 to learn the ins and outs of **object oriented** (OO) programming. Organizers had to change locations twice to accommodate the overwhelming response from financial houses either implementing or investigating this technology.

Nearly every major commercial bank, investment bank and brokerage firm sent developers who support front-office systems, with the largest contingents coming from Citibank, Chase Manhattan and Salomon Brothers. Others came from Smith Barney Shearson, Swiss Bank Corp., the Federal Reserve Bank, Moody's Investor Services, Standard & Poors, Reuters, Quotron and Dow Jones Telerate.

Not only was this the first meeting of OONY - the brand new users' group for **object-oriented** technology in New York - but it was also a chance to hear Adele Goldberg, co-founder of ParcPlace Systems, a spin-off of Xerox Corp.'s Palo Alto Research Center, discuss object-based project management.

This is a technology taking Wall Street by storm.

Benefits are the ability to code programs faster, lower maintenance costs and reduce the number of bugs that creep into financial software. Industry experts claim that OO is the only technology capable of keeping up with the increasingly speedy innovation of financial instruments, especially customized derivatives. 'Objects are going to enable the financial community to get their hands around the complexity (of financial instruments) for the first time,' says Patti Dock, a consultant and president of Pillar Systems Inc. in Sandy Hook, Conn.

That's because OO eases the process of dealing with rapidly evolving financial instruments by allowing the programmer to build an object library from which he can pull reusable chunks of code. Rather than perpetuate the tradition of coding each and every procedure from scratch, trading firms can encapsulate the properties of **financial instrument** components in discrete objects, which they can use over and over again.

Beyond the technical wizardry, can trading houses make money with OO? Yes, contend the software experts. 'Some of the most creative uses of **object oriented** technology are on the trading floor,' says Susan Cohen, a senioir analyst at Forrester Research in Cambridge, Mass. For one, the ability to rapidly strip in and strip out features to create exotic instruments increases a firm's competitive edge and likewise its profit margin. Secondly, once an object has been created and 'shelved' in an object library, its reuse should lead to a reduction of errors or 'bugs' in comparison to traditional methods. And fewer bugs should translate into lower maintenance costs.

In addition, OO is revolutionizing the role traders play in the design of new products and in the tracking of product lifecycles. Instead of waiting for programmers to design financial products, some traders can assemble new hedging strategies by simply pointing to objects labeled Interest Rates, Yield Curve, Cash Flows and Currencies. Once the technology has truly taken hold, experts see the chance for the integration between the front- and back -office functions. As functions normally associated with back-office work move onto the workstations, traders will take on a greater understanding of the full trading process.

But for all of the splendor of OO there are several hurdles the

financial industry must still face. Beyond the onerous learning curve, which experts claim is scaring off potential users, it requires a hefty up-front multimillion dollar investment as well as a modification of the way programmers traditionally approach projects. All this takes place in a relatively conservative industry comprised of traders who expect instantaneous results - whereas experts estimate OO users can wait anywhere from one to two years for anything tangible.

Previously, Wall Street churned out new products despite serious lags in technological advancements. Because firms were unable to process the most exotic products, the effects of the disparity could be felt in Wall Street pocketbooks. 'One of the most intractable problems for firms is the length of time it takes to develop applications,' says Cohen at Forrester. 'That gets in the way of firms competing effectively.'

To compete in the cutthroat world of custom derivatives, dealers must price, hedge and design transactions before software exists in order to book the business. 'The standard development techniques cannot keep up with the way the market is evolving,' says Jim Rogers, vice president of Sanwa Financial Products (SFP) in New York. SFP, the three-year-old fixed income derivatives trading subsidiary of Sanwa Bank, was in a unique position in the industry when approaching OO. As a newer entity, the firm began to explore OO 'from the beginning,' says Rogers.

Rogers admits SFP was hesitant to dive head first into OO due to the industry's lack of familiarity and dearth of knowledgeable programmers. But their need for effective technology was stronger. 'With derivatives you cannot create a system and walk away because by the time you design and implement it, it is out of date,' says Rogers, who chose Montage, object -based software from International Financial Technology (Infinity) Inc. based in Mountain View, Calif.

Today, almost two years after conducting a pilot project and deciding to go ahead with Sun Microsystems's C++ system in their derivatives trading environment, SFP has survived the onerous learning curve and has laid the groundwork for an OO environment. Rogers goes so far as to say OO gave SFP the opportunity to jump into the market faster. 'Once you build your infrastructure, it allows you to innovate quicker,' he says. 'The derivatives market is constantly changing, and that is where the **object oriented** market has tremendous value.'

Role Reversals. Along with a decrease in the turnaround time necessary for systems development, OO is affecting the traditional roles held by programmers. Programmers, who traditionally progressed up the corporate hierarchy and were compensated according to their project functions, are now considered 'designers' and 'implementers' in the new age of OO. And the differentiation between the two is somewhat sticky for human resources departments.

'The differentiation lines tend to go away,' says Dock at Pillar. Dock advises Fortune 50 companies on their strategies, organizational structure, roles and the eventual price tags in making the transition to OO. Dock attributes this change to the nature of the beast - OO demands that all players in a project have a hand in all stages of development rather than working around a multi-tiered programming structure.

While causing upheaval in programming roles, OO is also revolutionizing the way members of the financial industry traditionally view the makeup of financial instruments. 'From the technological perspective, **object oriented** technology changed the way we think about design,' says Michael Packer, managing director of Bankers Trust. 'It is a set of interrelated components rather than information that flows from box to box.'

Bankers Trust is currently utilizing OO in a joint venture with International Business Machines to build the LS2 system, a real-time commercial loan system. OO-based LS2 tracks the early discussion between the borrowers and lenders, the formation of deals, the primary syndication of the deals, as well as the automated funding of loans, servicing, secondary marketing and the trading of the loans, says Rich Freyberg, managing director of loan system development at Bankers Trust. Due to the fact that Bankers Trust is currently reengineering their commercial lending process, 'OO allows us to be flexible in the reengineering approach, i.e.,

we can change our approach,' Freyberg says.

OO is also prompting a change in trader functions. While there is still uncertainty as to the ultimate impact OO will have on traders, the revolution in software may spur a change for traders by placing traditional back-office processes on the workstation. 'OO has empowered the traders at the desks to do things that they used to have other people do for them,' Dock at Pillar.

Similarly, OO is prompting an evolution of the traditional trading environment, says Cedric Packham, vice president and director of information services at ScotiaMcLeod in Toronto. '**Object oriented** technology is allowing us to create some very effective front-office systems,' he says. Packham cites the capital markets trading desk Scotia implemented through Decision Software Inc. (DSI) of New York as an example. While Packham declines to attribute the move from back to front office solely to OO, he acknowledges that 'some of the functionality being provided in the front office, such as real time exposure and position management, is moving work from the back office forward.' Scotia's interest in OO led the firm to choose DSTS, DSI's real-time, multi-currency trading, position and risk management system. 'We were looking for a trading system that could take us through the mid-'90s,' says Packham.

While traders begin to reap the benefits of OO and create their own products via their workstations, a prospect that Philip Meese, director of technical services at Mercury Technologies in New York places 'in the long-term,' many questions emerge. What will happen to product innovation on Wall Street? 'I suspect you are going to see a lot more innovative products,' says Ron Dembo, CEO of Algorithmics in Toronto. And will traders require increased mathematical competency? Not so, according to Meese. Essentially, 'you can map the traders' world out,' says Meese. If OO keeps progressing, traders may eventually be able to develop highly structured deals based on elements such as cash flow, date series, yield curves and underlying instruments with which they are already familiar.

But Michel Hanet, vice president of trading systems at Chase Manhattan, is a bit more skeptical about the prospect of traders taking a more active role in the actual programming of new instruments. 'Traders won't develop instruments,' he says, 'unless it is layered in a user-friendly fashion.' Chase took on OO three years ago when it purchased Opus by Renaissance Software of Los Altos, Calif. for interest rate derivatives. Though the bank is keeping Opus for the foreseeable future to handle all of Chase's global risk management products, Hanet says the bank is now starting to use VisualWorks, the latest Smalltalk product by ParcPlace Systems Inc. in Sunnyvale, Calif. According to Hanet, VisualWorks will be used for the whole product spectrum at Chase. As an indication of the learning curve involved in OO, Hanet says that though the latest system was introduced to Chase five months ago, the firm is now 'barely seeing the changes.'

As for the rumors regarding the power of OO to actually erase the more extreme distinctions between front- and back-office functions, Dembo at Algorithmics tends to believe otherwise. 'I don't think it is integrating the front and the back office necessarily,' Dembo says. 'No one has really developed a good back office that is **object oriented**,' he points out. Nevertheless, Mercury's Meese claims sights for the future are set on the tasks of front-office and back-office integration.

Weighty Drawbacks. For all of its purported miracles, OO does have rather weighty drawbacks. For one, the prevailing opinion in the industry is that taking on OO means taking on a daunting learning curve - one that may have frightened away more than a few potential object users. 'Is the learning curve driving firms away?' asks Forrester's Cohen. 'Absolutely,' she says.

Experts estimate is takes anywhere between one to two solid years for programmers to successfully learn the ins and outs of OO. The same experts estimate the learning curve for fourth-generation languages (4GL) or C is six months. But, according to Dock at Pillar, in that one- to two-year period a programmer learns a lot more about the system than he would about 4GL or C - including design, debugging and testing.

But for all the stress and strain involved in learning OO, experts

claim the end results far outweigh any problematic roadblocks. 'If the goal of truly reusable [redacted]s libraries can be achieved, [redacted]is definitely worth the learning curve,' says Packham at Scotia. Dock concurs. 'If a person truly wants to learn **object oriented** technology, the learning curve is surmountable,' she says.

And, apparently, traditional programmers are on equal footing with those entering the market fresh when it comes to learning OO. The deciding factor is the individual's ability to grasp the new approach to programming. 'In the past, software was developed in a divide-and-conquer fashion,' says Cohen at Forrester. '**Object oriented** technology is a more holistic approach.'

Dock also pegs success in learning OO to the individual's attitude toward new technologies. When she conducted programmer training, 'success depended upon whether I dragged them in kicking and screaming or whether they came in on their own,' Dock says.

Scotia had a double-edged encounter with the OO learning curve, says Packham. In one sense, the firm did not experience the learning curve because DSI brought the OO-based trading system to the firm in production form, which eliminated the need for rigorous programmer training. But, to provide the firm with a better appreciation of the learning curve, two of Scotia's top programmers worked with DSI in the development stages. While one Scotia programmer readily grasped the concept of OO technology, the other programmer experienced difficulty. 'This experience showed us that the individuals' background and experience played a significant part in their ability to grasp the concepts,' says Packham. 'Programmers experienced in user interface more easily adapted to the OO environment.'

While the need for technological aggressiveness in learning OO programming is inarguable, Ray Dodd, a principle at Fusion Systems in New York, chooses to downplay the fear factor of the learning curve. 'There has been a tendency in the past to present **object oriented** technology as a new mystical technology where you have to forget everything that you have learned,' Dodd says. 'That is clearly bogus.' OO is based on previous platforms, knowledge and ways of doing things, Dodd says.

In addition to the arduous learning curve, a lack of technological OO experience in the financial industry and reportedly immature support tools have also raised concerns. 'A lot of people are not ready to go for it,' says Hanet at Chase. 'It is a major paradigm shift.' This reluctance has narrowed the field of experienced programmers - a necessary component for introducing the technology to a firm. The concern is rooted in 'the depth or lack thereof in the talent pool,' says Packer at Bankers Trust.

Out with the Old. Tying expectations for new technologies to the traits of old technology development is causing additional problems for OO's acceptance. First, traders are accustomed to tangible progress during product development. With OO 'you spend a lot of lead time developing infrastructure with very little to show,' says Meese at Mercury. 'Traders are used to having something to show.' But, again, experts harken back to the 'no pain, no gain' theory. 'One of the presumed benefits of **object oriented** technology is that if you do your homework upfront, you get a lot of benefits down the road,' says Dock at Pillar.

Firms must be patient and realize that the task of building an object library alone adds to the task. 'Until you get the objects on the shelf, you don't have the full benefits of the technology,' says SFP's Rogers. 'In developing objects you are paying your dues up-front and investing in technology.' Meese at Mercury agrees. 'There is an investment in infrastructure but once the infrastructure is there the applications are rapidly developed,' he says.

Second, Dock prefers to play up OO's role in empowering individuals in the development process, whereas tools were about removing some of the more mundane aspects of human involvement. 'It is not about control, and CASE (computer-aided software engineering) tools were about control,' she says. Dock's concern is the industry's search for tools similar to those for older technologies - tools that may never exist. Regardless, Dembo at Algorithmics says even tools such as compilers for OO still have a long way to go. 'The basic tools are still flaky,' he says.

Overall, budgeting time and money for conversion to OO is a function

of firm size and what applications the firm is seeking to develop, says Hanet at Chase. Financial investment in OO depends on several factors, including whether firms wish to convert all processes to OO or just specific projects and whether firms bring in outside consultants for the transition, says Dock. At the least, large corporations will need to invest millions, 'not hundreds of thousands,' each year in OO, she adds.

Hypothetically, if a firm is conducting three pilot projects, Dock estimates that the firm could spend in the neighborhood of $180,000 on consultants as well as another $54,000 training internal personnel to be 'mentors' on later projects. The other possibility is for a firm to send select internal personnel to an off-site apprenticeship program, which Dock estimates could cost $80,000 per project team. This is all in addition to training for the remaining internal personnel to acclimate them to OO as well as to software costs, which generally depend on a firm's deal with a vendor. 'Every firm has very different and personalized goals and objectives in mind when transitioning to objects,' says Dock. Regardless, after the first year a firm's OO costs will 'absolutely decrease,' says Dock, because the first year is spent training people.

So where will OO go from here? 'OO is still coming of age,' says SFP's Rogers. Likewise, Packer at Bankers Trust says there is 'a tremendous amount of evolution to come' with regard to OO. But there is the inevitable fact that all technologies eventually gather dust. Despite his declaration that '**object oriented** technology is a pretty big evolutionary step,' Dembo at Algorithmics remains steadfastly realistic about the future. 'Within five years you will see another paradigm or some major enhancement,' he says.

But for now things are looking up for OO. Says Dr. Jeffrey McIver, director of financial engineering at Infinity: '**Object oriented** technology won't be the new kid on the block. It will be the only kid.'

PUBLISHER NAME: Miller Freeman, Inc.
EVENT NAMES: *600 (Market information - general)
GEOGRAPHIC NAMES: *1USA (United States)
PRODUCT NAMES: *7372411 (General Accounting & Financial Software)
INDUSTRY NAMES: BANK (Banking, Finance and Accounting); BUSN (Any type of business); CMPT (Computers and Office Automation)
NAICS CODES: 51121 (Software Publishers)

4/9/23 (Item 8 from file: 16)
DIALOG(R)File 16:Gale Group PROMT(R)

03014382 Supplier Number: 44093142 (THIS IS THE FULLTEXT)
Prometheus Unplugged
Forbes, pS149
Sept 13, 1993
ISSN: 0015-6914
Language: English Record Type: Fulltext
Document Type: Magazine/Journal; General Trade
Word Count: 1240
TEXT:
BY JEFFREY YOUNG

DWIGHT KOOP WORKS for one of the world's largest, and most conservative, Swiss banks. As he attends a meeting in an elegant conference room, the beep from the pair of palm-sized black boxes on the table in front of him is barely audible. Nestled in a leather carrying case and held in place by Velcro fasteners, the Hewlett Packard 100LX, a top-of-the-line programmable calculator-cum-palmtop-computer, is tethered to an Ericsson GE Mobidem wireless modem whose flexible plastic antenna is waving in the breeze from the air conditioning. Both devices are battery-powered. A symbol atop the modem's LCD screen flashes on and off every few seconds, indicating that something is coming in, or going out.

That's not the only thing going on.

In the meeting room, the voice of an executive in Switzerland booms out of the speakerphone. Beyond the glass wall, a mezzanine gallery circles a basketball-court-sized three-level space filled with pods of desks, stacks of computer monitors and the requisite wall of moving ticker symbols, stock prices and news feeds. This is the trading floor of the Swiss Bank Corp. in Chicago, four floors above the chaos of the Chicago Board of Trade's commodities futures pits. Here, in contrast to the frenzy of colored jackets and hand gestures in the pits, all the action happens on Next, Sun Microsystems, Hewlett Packard, Symbolics and other workstations - as well as on a growing number of handheld, wireless devices of all kinds.

This afternoon the group in the meeting room is discussing new software developments that Swiss Bank is planning. But Dwight Koop, 45 years old and executive director of information technology for Swiss Bank Corp., has other things on his mind. A few taps on the tiny keypad move him through a list of 15 E-mail messages that have landed in his wireless mailbox with the latest beep. The HP 100 and modem are small and unobtrusive enough that he can use them without disrupting the meeting. He stops at one message, pulls it up on the screen and squints at it through his glasses. A faintly worried look passes over his face. A few more taps and a reply shoots out into the ether.

Simultaneously, his SkyTel SkyWord beeper starts vibrating. Koop checks it, then punches his watch, a Dick Tracy-like digital artifact that not only tells the time but also is a paging device. He stands up, closes his equipment, mumbles something about 'putting out a fire' under his breath so the telecommuting executive can't quite hear him over the speakerphone, and breaks away from the meeting.

Koop loves the gadgets. But he's also charged with integrating them into Swiss Bank's work culture, and that, he says, won't be easy: 'It's not all ready for prime time yet. The service's interface and interaction model is troubling. For someone who has spent years messing with computers, it can be navigated. Far too often I have to fight it (the system) to make it work.'

Koops says wireless trading will 'languish' until somebody figures out all the pieces and integrates them. Today Swiss Bank sends its traders onto the floors of the exchanges with print-outs from its internal computer systems. The bank would like to someday equip its traders with handheld machines. 'It won't be anytime soon,' Koop says. 'Trading is a contact sport.'

Koop's job involves keeping the bank's global trading systems up and running. These are not just any trading systems. Over the past few years, Swiss Bank has acquired the majority of Chicago's O'Connor Partnerships (the final portion of the acquisition is awaiting SEC approval). O'Connor is one of the most highly secretive, technically sophisticated and profitable of the world's trading operations dealing in financial derivatives - products like options and index instruments that are derived from underlying traded securities such as stocks, bonds and currencies. A leader in the application of mathematical algorithms to options, currency and **financial instrument** trading, O'Connor was once known for trying to avoid all publicity. It once shreded the packaging materials for the workstations it bought and required all employees to sign extensive secrecy agreements. Even today the closest a visitor can get to an O'Connor computer screen is the gallery, a good 30 feet from the trading floor.

Creating new bundles of instruments and options to sell in response to customer requests - instantly - is a growth opportunity that savvy traders have been eager to exploit. All of this is much better handled by computers, which O'Connor realized in the mid-1980s when it led the financial community in moving first to Sun machines and later to more sophisticated workstations. But the new trader's edge is literally up in the air.

WALKING AROUND - GLOBALLY

If any trading company figures out wireless, it probably will be Swiss. Swiss Bank likes to push the techno-envelope, for example, buying 500 Next machines a few years ago because it believed in the future of **object-oriented** development systems. 'We have no idea if Next will make it in the long run,' says Koop's boss, Craig Heimark, Swiss

Bank's managing dir[redacted]or of technology. 'But someone will succeed with objects. And we'll [redacted]eady have a wealth of experien[redacted]programming in this kind of environment.'

Koop enjoys explaining that he has signed 425 nondisclosure agreements with technology companies. 'When companies ask if they can qualify us as a beta site,' he says, 'I explain that they don't seem to understand. We qualify them.' This is real money being handled - security is a big issue. 'Dial-in modems simply don't exist in our world,' he adds quietly.

For all of O'Connor's history of secrecy, Swiss Bank is talking about moving its proprietary risk-management trading systems to customer sites. The idea is for the company to let customers manipulate versions of its proprietary analysis tools, then sell them the financial products that meet their needs. Enter wireless. 'We want our managers to work by walking around - globally,' Koop explains. 'But we sure don't want these top-level people fumbling for the phone jack and negotiating with the PBX operator to get a dial-out line so they can check the day's market.'

For all his complaining, Koop is still certain that wireless will be a key communications component over the next few years. His own love of the toys convinces him.

On another day Koop is sitting in a hotel in Sausalito, at a window overlooking San Francisco Bay and the city skyline. His computer and modem are open in front of him, and the radio signal is strong. The indicator is flashing furiously. His attention is intermittently pulled to the screen. It is more than a little disconcerting to try to talk with him. We may have to learn a new social dynamic in the era of ubiquitous wireless machinery.

Suddenly he chuckles, and pushes the unit over to display a message he's just received. 'A bunch of us have an unofficial contest to send a message from the most unusual location,' he says. 'Have you ever heard of this place?' The message is tough to see in the glare, but when the tiny LCD display is positioned correctly, it reads: 'Do I win the contest? I'm sitting at the bar of the Jaguar Club in Atlanta.' (The Jaguar Club is one of the more infamous topless bars in the South.) Ah, brave new world. As it turned out, that message didn't win. The winning one was sent from atop Elvis' grave at Graceland.

PUBLISHER NAME: Forbes, Inc.
COMPANY NAMES: *Swiss Bank Corp. (Switzerland)
EVENT NAMES: *260 (General services); 330 (Product information); 360 (Services information)
GEOGRAPHIC NAMES: *1USA (United States)
PRODUCT NAMES: *6231106 (Chicago Board Options Exchange); 3662116 (Wireless Local Area Networks); 3661271 (Data Modems); 4811500 (Specialized Telecommunication Services); 4838012 (Satellite Paging Services); 6020000 (Commercial Banks)
INDUSTRY NAMES: BUS (Business, General); BUSN (Any type of business)
NAICS CODES: 52321 (Securities and Commodity Exchanges); 33422 (Radio and Television Broadcasting and Wireless Communications Equipment Manufacturing); 334418 (Printed Circuit Assembly (Electronic Assembly) Manufacturing); 51331 (Wired Telecommunications Carriers); 513321 (Paging); 52211 (Commercial Banking)
SPECIAL FEATURES: INDUSTRY; COMPANY

4/9/24 (Item 1 from file: 148)
DIALOG(R)File 148:Gale Group Trade & Industry DB

09833050 SUPPLIER NUMBER: 18517501 (THIS IS THE FULL TEXT)
The case for expressive systems.
Pawson, Richard; Bravard, Jean-Louis; Cameron, Lorette
Sloan Management Review, v36, n2, p41(8)
Winter, 1995
ISSN: 0019-848X LANGUAGE: English RECORD TYPE: Fulltext; Abstract
WORD COUNT: 6005 LINE COUNT: 00500

ABSTRACT: A new [redacted]d of information system is eme[redacted]ng that will reduce the time to market, help tailor products and services to customers' needs, and make processes more responsive to unexpected events. Expressive systems allow users to adapt quickly and easily to exceptions from standard operating procedure. The authors describe how expressive systems work and suggest ways of modifying the roles and structures of IS departments to implement the new technology. (Reprinted by permission of the publisher.)

TEXT:

On the derivatives trading floor at J.P. Morgan in New York, there is a new information system called Kapital. The trading environment supported by this system is a demanding one: the traders who use it are at the cutting edge of creativity in the financial markets. In addition to trading a wide variety of instruments, they are continually inventing new instruments by combining parts in new ways, "bundling," or fashioning an entirely new set of custom terms and conditions. Conventional applications development approaches do not easily support the degree of systems flexibility required by such an environment.

Kapital employs some of the most sophisticated technology currently fashionable in the world of information systems today: it is based on a distributed client/server architecture, borrows techniques from the world of artificial intelligence, and is one of the purest implementations of the concept of **object-oriented** software in the business community. Using Kapital, traders can choose the user interface that suits them, from simple business forms to graphical representations. They can perform powerful financial analytics using complex mathematical models. Kapital accommodates real-time data feeds giving current market information and performs sophisticated portfolio analysis against a variety of market assumptions.

However, what really distinguishes Kapital from other information systems is not the technology, but the fact that it does not attempt to fulfill a specified set of user requirements -- at least in the conventional sense. Rather, Kapital attempts to model the very "language" of J.P. Morgan's trading business -- not only the vocabulary, but also the grammar and, arguably, the style. Kapital implements that language in software, so that the traders can directly express their ideas for new "exotic" tradable instruments. One objective for the system was that, as fast as traders could conceive a valid opportunity for a new kind of **financial instrument**, he or she should be able to directly interact with the system to create that instrument, simulate its performance in terms of risk and profitability, and if satisfied, price and trade it immediately. To do this required that the software present atomic-level financial components and business rules to the traders, along with the capability to manipulate and extend them m new ways.

Kapital is not a programming language in the conventional sense; it bears no resemblance to the so-called "fourth generation" programming languages on the market today, nor even the graphical programming tools now gaining popularity. Its basic constructs are not simply high-level representations of the computer's resources (although these are provided where useful), but the natural constructs and components of the trader's world: cash flows, interest rate scenarios, risk profiles, and so forth.

While giving users direct manipulation of the system's capabilities, Kapital does not eliminate the need for professional programming. A high-caliber team needs to maintain and continuously enhance the business language and its supporting technologies. Furthermore, traders may call on professional programmers to assist them in implementing a more difficult idea, refining a prototype they have developed, or writing a new component from scratch. However, the response time for such requests is measured in minutes and hours, not weeks and months. In part, this is because Kapital provides high-level business capabilities and low-level technical components in one integrated environment. The developers do not have to translate a requirement from a business domain, as with conventional programming, into a different technical medium.

Kapital is an example of a new kind of information system that is

beginning to emerge in business, which we have dubbed "expressive systems." Conventional systems support only the standard business operating procedures for which they were designed.(1) Expressive systems are designed to support exceptions from standard operating procedures, empowered actions, new product ideas, services tailored to individual customer's needs, ad hoc or even temporary changes in organizational structure -- none of which, by definition, can be specified up front. These changes Call be implemented in a time frame appropriate to the business need -- in some cases, in seconds or minutes, by the users themselves; in more complex cases, in hours or days, with the help of professional programmers; but never in terms of weeks or months.

Expressive systems not only make it easy to implement these changes, they encourage business managers or empowered workers to explore more possibilities for change and thus improve business performance. Using an expressive system within any of these contexts feels as natural as using an electronic spreadsheet program to develop a financial model and then explore "what-if" scenarios.(2) But expressive systems are not mere simulation tools. They permit the user to execute the action through the same system, whether that result is a new **financial instrument** to trade, a new way of routing documents through an administrative function, or a reallocation of work orders between manufacturing plants.

In this paper, we show, with reference to both examples and new theory, that expressive systems will in time become the dominant paradigm for business computing. Implementing expressive systems, however, will require organizations to address new technologies and redevelop many of their existing systems. Furthermore, the implementation and support of expressive systems will require a substantial modification to the roles and structure of the information systems department.

Roles for Expressive Systems

We have identified three specific roles that expressive systems will play in business:

* Reducing the time to market for introducing new products.

* Facilitating the tailoring of products and services to individual customer's needs.

* Making operational processes more responsive to unforeseen events.

(Interestingly, these three roles correspond to Treacy and Wiersema's three dimensions of market leadership: product leadership, customer intimacy, and operational excellence.(3) This suggests that expressive systems have a role in all organizations that seek to lead their markets.)

Reducing Time to Market

In many industries, from automobiles to pharmaceuticals, reducing the time to bring new concepts to market is critical. Computer-aided design systems, arguably an early form of the expressive system concept, have encouraged users to explore more design alternatives and better understand the consequences of their actions. The new Boeing 777 aircraft was completely designed on computers using a package called Catia. For the first time in modern aircraft design, it was not necessary to build an evolving full-scale prototype to find out if the pipework and other systems could be routed through the narrow confines of the airframe. The computer simulation provided that intelligence -- the first 777 was built to fly.

The advance of new prototyping technologies, such as stereo lithography, which can create a plastic three-dimensional form from a computer model in seconds, and robotic assembly means that the users of such systems can express their ideas directly into physical form. However, the greater the information content of products and services, the stronger the potential for expressive systems to reduce the product development time.

In a limited range of cases, expressive systems can potentially eliminate the product development process altogether. Arguably this is the case with J.P. Morgan's Kapital system. The ability to create new financial instruments "on the fly" changes the trading paradigm from looking for opportunities to trade each of a set of preexisting instruments to creating the instrument needed to take advantage of each opportunity that arises.

While the financial trading environment is undoubtedly a rarefied one, it is worth noting that there is a clear pattern of innovations

transferring from this environment to "ordinary" business.(4) Telecommunications companies and energy utilities, for example, are starting to deploy systems of similar sophistication in order to introduce new forms of billing and new value-added services, in response to rapidly changing market conditions.

Tailoring Products and Services to Customer's Needs

Many organizations recognize that to retain their most valued customers, they must increasingly respond to an individual customer's needs. The difficulty lies in being able to do this with something approaching the economy of standardized services; information systems are clearly the key to this objective. Banks and other financial services have for years been developing "parameter-driven" information systems that would allow them to vary the parameters of a financial product (the interest rate, term, and any discounts) without the need to write new program code. In reality, however, the degree of customization this approach offers is still small, and the professional systems effort needed makes it uneconomical for application to individual customers. One organization that is seeking to break this limitation is Britain's National Westminster Bank (Nat West).

As part of a series of initiatives to define the future of retail banking, the IT strategy department at Nat West developed sophisticated PC based software that would permit not only the parameters, but the very operating structure, of a bank account to be tailored. Suppose, for example, that a high net-worth customer preferred to have her checkbooks sent, not to her home, but to the nearest branch of the bank -- with an advisory letter sent to her home. This apparently simple requirement would be beyond the scope of most parameter-driven systems and would require expensive manual intervention. Nat West's prototype system permits this scenario simply to be drawn out graphically on the screen, creating the necessary supporting systems automatically.

Several things about Nat West's prototype system stand out:

* The software was designed to be used by an IT-literate bank manager or, more probably, by a systems professional sitting next to the bank manager. But, either way, the new type of account is created in real time, typically m response to a customer request.

* The system is graphical, with icons representing all the business constructs that a bank manager would expect to deal with -- customer, interest rate calculator, statement, checkbook, and so forth.

* The system does not present the user with a series of predefined options. Rather, it feels like a well-designed child's construction kit, with which the user can quickly explore and implement almost any idea.

* The user can see the profitability, or otherwise, of the financial product. (This can even be done on a separate screen, allowing the basic product to be designed in front of the customer.)

* Built-in constraints prevent the user from building illegal or nonsensical financial products.

It may be several years before this scenario pervades the bank's branch network, but Nat West is committed to implementing its system and probably has at least a couple of years' lead on its competitors. The difficult part is not designing the graphical front end, but redesigning the deep structure of the core information systems to allow them to be manipulated this way.

Expressive systems therefore perform two important functions in product or service customization. The first is to permit the customization to take place in "real time" at the point of customer contact, rather than later in a back office. The second is to permit the user tO explore and understand the consequences (here, in terms of profitability) of the proposed approach or action. Without this capability, there can be no substance to the concept of empowered actions. This theme carries over into the third function.

Responding to Unforeseen Events

In the context of improving operational performance, expressive systems take on two roles: the first is to optimize the refinement of resource use; the second is to facilitate the handling of unexpected events and potentially chaotic disruption.

Manufacturing, logistics, and other key operational functions have been subject to intensive study and refinement for many years. Both quality and efficiency have been driven up, waste reduced, and nonproductive costs, such as stock, virtually eliminated. The scope for further refinement is narrowing rapidly. But the refinement has brought a new problem: greater potential for chaotic disruption (we use chaotic in the mathematical sense).

American Airlines, whose operating procedures and information systems are second to none, has recognized this. Its systems operational control is the business unit charged with executing the flight schedule and marshaling the many different resources on which it depends. Any flight may draw its plane, flight crew, and cabin crew from three different incoming flights, while baggage handling, gate space, catering, cleaning, and other ground-based resources must also be coordinated. With this level of dependency, unforeseen events, from unscheduled maintenance to adverse weather (not to mention presidential haircuts!), have the potential for chaos. Currently, the antidote means preserving the structure of the dependencies at all cost -- even if this means delaying whole complexes of flights. No one really knows the true cost of these off-schedule operations because of the difficulty of separating the complex costs from routine operations and estimating the impact on customer loyalty, but they are believed to be enormous.

Reducing this cost cannot, by definition, be achieved in the same manner as previous operational refinement. As part of a long-term program to apply the power of IT to this thorny problem, American Airlines has developed new systems specifically to support its flight dispatchers, the individuals who manage a flight from the ground, including all resources and any route changes. Previously, flight dispatchers had to access information systems through the same kind of transaction-oriented interface as the reservation systems. The new system not only provides more direct manipulation of the system, but also helps the dispatchers to explore the consequences of each unscheduled event and each possible response. One feature of the user interface, for example, is a time line that graphically portrays the prior events on which a particular flight depends, future flights that in some way depend on it, and their current status. Dispatchers can instantly see the consequence of shifting the proposed take-off time, in terms of disruption to the schedule, to staff and internal resources, and, of course, to passengers. No airline currently has an effective model of the financial costs of such disruption, or of the impact on future revenue from customer dissatisfaction, but the American Airlines system is a significant step in that direction. One way of looking at this type of expressive system is that it takes the conventional concepts of operations research (including, for example, PERT networks) and makes them an integral part of real-time operational systems.

A second example is Black & Decker (B&D), whose complex manufacturing operations have been based for twenty years on MRP II, the standard for manufacturing resources planning. An increasing proportion of B&D's sales is coming from large and streamlined retail operations, such as Home Depot, that may place orders on the basis of "deliver within seven days or the order is canceled, and we devote the shelfspace to competitive products." With some retail stores four days away by road, this gives B&D just three days to respond. But the MRP II systems require so much setting up and processing time on mainframe computers that they can be run only every seven days.

Moreover, the MRP II algorithm works in one direction only: it converts a master build schedule into a materials requirements plan and thence into a capacity plan. If there is an unexpected change to the availability of manufacturing capacity, or to the availability of materials, the MRP II system cannot identify the consequences or advise alternative actions.

Against this backdrop, B&D formed a new team for advanced manufacturing technology with the goal of designing the manufacturing control system of the future. That system, built with the help of a small but innovative software vendor called Intellection, is now in operation in several of its plants. B&D believes that it now has an operational

flexibility that not even the Japanese can match. It allows the company to consider the consequences of any event and dynamically explore alternative ways to meet the same requirements. In the not-too-distant future, the system will be accessible from every machine cell in the plant, enabling individual machine operators to identify and understand the consequence of, say, shutting down the machine for an hour's preventative maintenance and finding alternative ways to get the parts made in the meantime.

Thus a key role for expressive systems in high-performance operations is to reduce the potentially chaotic effect of disruptive events. Henry Mintzberg wrote, "When the planners run around like Chicken Little crying, `The environment is turbulent! The environment is turbulent!,' what they really mean is that something has happened which was not anticipated by their inflexible systems."(5)

Expressive Systems Contrasted with Other Systems

There are other kinds of information systems, such as decision support systems, and other new approaches to systems development, such as rapid application development, that are seeking to address some of these same business goals.(6) However, there are also some clear distinctions, and it is these distinctions, we believe, that make the expressive systems approach more effective in meeting those goals.

Decision support systems, or executive information systems, have made it possible for users to express their information requirements directly, and their ease of use has encouraged managers both to analyze past performance in greater depth and to simulate better the possible consequences of proposed actions.(7) However, the functionality of executive information systems is typically limited to obtaining and analyzing information -- they are not a medium through which actions can be executed, in contrast to each of the examples discussed above.

Furthermore, creating a decision support system is usually a matter of grafting new software on to the front end of existing transactional systems. The front end, whether an off-the-shelf package or specifically designed for its purpose, shields the user from the technical details of accessing the underlying systems, provides powerful data manipulation tools, and typically wraps the whole in a nice graphical interface. Newer generations of such packages permit the user to change data stored in the underlying systems, for example, to change a customer's address, and perhaps to invoke standard procedures, such as "issue a statement" without leaving the graphical environment. But the only actions or functions available to the user are the fixed set of transactions that the underlying system was designed to support.

If expressive systems are to support actions not previously specified, the user needs access not only to predefined transactions, but to the building blocks from which new kinds of transactions can be constructed. Most of today's core transactional and other "mission critical" information systems were not written with this objective in mind and do not support access at this component level. Some very modern, large systems include application programming interfaces (APIs), which provide better access to the underlying functionality, but these are intended for professional programmers who will be constructing substantial new software applications. Creating the component level access required to implement the expressive systems concept typically requires a complete rewrite of existing core systems.

Within the IS community, the biggest competitor to the concept of expressive systems is probably rapid application development (RAD)(8) -- the new techniques for dramatically reducing the lead time to develop new systems. Some instances of RAD deploy similar technology to that in expressive systems (including client/server and object orientation). Some use conventional systems technology and programming languages but deploy different management techniques such as intensive workshops involving users and developers (sometimes called joint application development, or JAD), and time-boxed deliverables.

The significant difference between the RAD approach and the expressive systems approach lies not in the technology or the actors, but in the intent of the process. JAD and RAD are fundamentally techniques for getting better agreement and ownership of the required specification,

either through intensive user/developer workshops at the start of the process, or through an iterative process of delivering crude prototypes of systems and refining the specification based on user feedback from those prototypes, toward a stable end point.

In the expressive systems approach, the iteration is primarily away from a stable start point. The basic components and capabilities of the system provide the start point, but as individual business units or individual users use them, they will move away from the start point as their needs change.

Implementing Expressive Systems

Expressive systems therefore change the concept of an application. Today, the term "application" refers to a collection of programming code and data that together meet a neatly circumscribed and well-defined set of business requirements, such as an order-processing system or a credit management system. Applications account for the greater part of the budget, manpower, and management attention of most is departments. The applications are supported by a common technical infrastructure, whose costs and management are shared, but these are smaller by proportion. Implementation of an expressive system requires an inversion of this balance, with a much thinner applications layer and a much thicker (or richer) shared infrastructure, which comprises not only technical components but also business constructs.(9)

If they are thin enough, applications can be thought of as a wiring layer. We find that this notion has particular appeal to systems professionals old enough to remember analog computers, in which applications were constructed by wiring together standard components on a plug-and-socket panel. Moreover, some of the PC- and workstation-based software tools most appropriate to building expressive systems (for example, Digitalk Parts, NeXTStep, and IBM's VisualAge) permit software components to be visually wired together on screen.

We could go farther and say that the word "application" changes its sense from a noun to a verb (strictly, the gerund of a verb).(10) Application now refers to the process through which the infrastructure is applied to the needs of a business situation or an individual user, rather than to a piece of software in its own right.

So what does the thicker infrastructure actually contain? The sine qua non for expressive systems is an "uncommitted" software model of the business (or the particular domain of the business that the system is to serve). In microelectronics, an uncommitted array is a silicon chip that is made as a standard component but, in the final stage of manufacture, is committed to a specific customer application, such as the video circuitry for a games machine or the ignition control system of a car. Similarly, an uncommitted software model represents a business in generalized form but can be committed (or recommitted) to a particular product set, market channel structure, or business organization.

For some years, there has been limited implementation of this concept in the form of tailorable software packages, which are now gaining in popularly. Here, the user organization has the ability to choose between a number of options (possibly a very large number), predetermined by the vendors of the package. In a truly uncommitted software model, however, the designers have not attempted to foresee all the possible configurations. It is like the difference between a model car that can be customized with decals and accessories, and a Lego set.

Designers of children's construction sets face a constant trade-off. Versatiliy requires more different kinds of components and lower-level components (individual wheels and bricks). Ease of construction demands fewer components, and this typically translates to ready-made subassemblies, like a vehicle cab or house roof. Designers can partially overcome this by creating powerful high-level components that take on several different roles. This principle is called "abstraction." and it is the key to building powerful uncommitted software models of the business. The following example illustrates why:

Three years ago, the Bradford & Bingley Building Society (roughly equivalent to a savings bank) replaced most of its systems portfolio with a new system, developed from scratch and based on an uncommitted business

model. The old system was proving increasingly costly to maintain and needed replacement anyway. The principal objective and the reason for the choice of approach, was a system that no longer constrained innovation. The chief executive himself stated that he did not want to be told that he could not implement an organizational or product change because the system would take two years to modify -- a clear, if negatively stated, call for an expressive system.

Within Bradford & Bingley's system is a generic products engine, which is built around such a generalized concept of a savings product chat it is capable of also serving as a loan or insurance product. Each product is attached, not to a customer, but to a more abstract software construct known as "associate" -- defined as a party with which the firm has a relationship. More specific versions of associate include the conventional notion of customer, but also agents and branches (retail outlets). By making all other parts of the software interface with the abstract or common version of these specific entities, a decision to change an agent-based product to a branch-based product has no external impact. Equally, if the firm decided to launch a specialized savings product for its own employees, which might entail special terms or security arrangements, it merely requires a new specialized subclass of associate called "employee," but no change to the product engine. Bradford & Bingley's system also has generic software engines for document creation, for the management of workflow in an administrative process, and for managing selected groups of associates for marketing.

Abstraction is not a new concept in information systems; indeed, it is a basic principle of data modeling, but there have been few attempts to apply this to the functionally (i.e., the code) of systems. Historically, the view has been that code needs to be written to meet the specific needs of an individual application. To the extent chat there has been any reuse of existing code, it has been at a very technical level: reusable subroutines for implementing a complex mathematical function or for managing computer resources. Little attention has been given to identifying generic or abstract business functions and implementing these as reusable components.

The advent of object orientation is changing this by replacing the artificial separation of code and data with the more natural concept of self-contained objects. A software object completely models a component of the business domain: it contains the data that represents that component and all the functionality that may change or interact with that data.

Object orientation has a natural fit with expressive systems in several ways: it underpins most sophisticated graphical user interfaces and facilitates the construction of reusable components. However, the greatest significance of object orientation, and the one least understood by most IS professionals, is its ability to support business abstraction. Two principles of object orientation apply here. The first is called inheritance, which facilitates the creation and management of specialized subclasses of objects, as in the relationship between associate and customer. The second is called "polymorphism," in which different objects execute different code in response to the same message. A spreadsheet and a word-processed document can both respond to the message "print," but the way they perform that function will be different. Polymorphism simply reflects the reality that, in business, there are fewer things we want to do than ways we want to do them. Expressive systems should provide the support for the generic things we want to do, and the user's application of that capability implements the particular way it is to be done.

The New IS Department

What kind of IS department will be needed to implement and/or support information systems that are based primarily on the expressive systems model? No single organization that we have encountered has yet completed this transition, but those who have partially moved toward expressive systems have had enough experience that we can piece together a plausible model for the future IS department. Organizations that identify with the potential benefits of expressive systems must recognize that implementing the concept is as much about changing the structure and behavior of the IS department as it is about changing the technology.

The first distinction between this future model and the IS department of today is in the realm of values, attitudes, and beliefs. Take the widespread belief that "If only we could get the users to specify exactly what it is they require, then our problems would be solved." The concept of expressive systems, unlike iterative development, is not based on the notion that users have difficulty expressing their exact requirements; it is based on the realization that increasingly users cannot state their requirements completely because they themselves cannot know all the business conditions and events they will be facing.

A second distinction concerns the role of systems themselves. In the future model, the role of systems is primarily to facilitate change. This means that the IS department must anticipate business change. It does not mean that IS must predict business change; rather it means that it must build systems that are resilient to future change through the use of abstraction, componentization, and rewirable infrastructure.

Thirdly, IS professionals must change their current beliefs about the difference between developers and users. The distinction has been historically valid because the user and developer dealt with different views of the same system. The developer's view comprised lines of programming code, data structures, and operating system calls, which together form a high-level representation of the computer's resources. The user's view comprised menus of commands, forms to be filled, queries and reports, which together form a representation of the specific business operations being supported. It is the translation between these two representations that makes conventional systems development such a time-consuming, arduous process. Expressive systems resolve the problem by having the developer and the user share the same representation -- a representation of the natural components and structure of the problem domain rather than a specific solution to it.(11)

Consider, for example, a spreadsheet. Its success lies in the fact that its constructs (tables, cells, and formulas) are a very natural representation of the structure of financial modeling problems and are suited to both very simple and very complex applications. If you have used a spreadsheet, you have almost certainly developed an application; moreover, between developing the spreadsheet and using the resulting application, there is no switch in the representation used. There is a clear distinction between the authors of the spreadsheet package (Microsoft or Lotus, say) and the user/developers, but this is not the same relationship as between conventional systems developers and users.

It is a curious fact that while many IS departments acknowledge the very sophisticated spreadsheet applications within their businesses, few provide any real support for spreadsheet development. Indeed, there is often an underlying cynicism with regard to end-user development in general. Professional developers are fond of quoting surveys demonstrating that 70 percent of all spreadsheets contain some kind of error but are less keen to help reduce those errors.

In the expressive systems era, professional developers still have roles to play, but those roles will change. Some will be deployed in the creation, management, and continuous enhancement of the infrastructure. This, we believe, will divide into three processes: "Business model maintenance" will be concerned exclusively with the uncommitted software model. Its developers will be high-caliber abstract thinkers, and a key issue will be the communication of the knowledge of this model to those applying it. The second process is concerned with the technologies that support the business model. One of the keys will be ensuring that significant new technologies are introduced to the system in the form of generic infrastructure services, rather than as specific applications. The final process we might call technology services, which most closely resembles the IS operations function today, except that it will be concerned with monitoring and improving performance at the level of individual software components rather than just of whole systems.

The professional developers' other role will be to act as mentors within the application process.(12) For example, in the sales and marketing department of Clorox Company, developers have completely adopted this role. In 1985, Clorox installed one of the earliest, truly expressive data

retrieval and manipulation packages, Metaphor (which was also an early example of an **object-oriented** system). In Metaphor, users write modules or capsules and then visually wire the capsules together to generate the reports or screens they require. Almost 150 people in the sales and marketing department use Metaphor intensively, and they are supported by between one and three systems professionals, as needs vary. As part of their mentoring role, the professionals look for commonality in capsules written by different users. The professionals rewrite those versions into a standard, robust, more flexible capsule and offer it around to all the users (who themselves often share capsules with each other via e-mail). The notion of improving user-developed systems would be anathema to many systems professionals. But Clorox has found that the net ratio of functionally created to professional systems input exceeds every other application of information systems in the company -- which more or less sums up the case for expressive systems.

References

(1.) E. Dyson, ed., "An Explicit Look at Explicitness," Release 1.0, October 1993, pp.1-19.

(2.) B. Nardi, A Small Matter of Programming (Cambridge, Massachusetts: MIT Press, 1993).

(3.) M. Treacy and F. Wiersema, "Customer Intimacy and Other Value Disciplines," Harvard Business Review, January-February 1993, pp. 84-93.

(4.) T. Malone, J. Yates, and R. Benjamin, "Electronic Markets and Electronic Hierarchies," Communications of the ACM 30 (1987): 484-497.

(5.) H. Mintzbeig, Mintzberg on Management (New York: Free Press, 1989), p. 243.

(6.) S. Haeckel and R. Nolan, "Managing by Wire," Harvard Business Review, September-October 1993, pp.122-132.

(7.) J. Rockart and D. De Long, Executive Support Systems (Homewood, Illinois: Dow Jones-Irwin, 1988).

(8.) J. Martin, Rapid Application Development (New York: Macmillan, 1991).

(9.) R. Pawson et al., Building the New Information Infrastructure (London: CSC Foundation, 1993).

(10.) R. Morison, The New I/S Agenda (Cambridge, Massachusetts: CSC Research and Advisory Services, 1994).

(11.) J. Tibbets, "Object Orientation and Transaction Processing Where Do They Meet?" (Phoenix, Arizona: OOPSLA 91, Sixth Annual Conference, addendum to the proceedings, 6-11 October 1991), pp. 3-15.

(12.) B. Nardi and J. Miller, "Twinkling Lights and Nested Loops: Distributed Problem Solving and Spreadsheet Development," International Journal of Man-Machine Studies 34 (1991): 161-184.

Richard Pawson is European director of research for CSC Research and Advisory Services. Jean-Louis Bravard is a managing director at J.P. Morgan & Company. Lorette Cameron is a vice president at J.P. Morgan.

INDUSTRY CODES/NAMES: BUS Business, General; BUSN Any type of business
DESCRIPTORS: Information systems--Evaluation
FILE SEGMENT: MC File 75

4/9/25 (Item 2 from file: 148)

09185984 SUPPLIER NUMBER: 18954598 (THIS IS THE FULL TEXT)
Enterprising solutions. (enterprise-wide risk management for financial institutions)
Shefi, Hagay
LatinFinance, n82, p73(2)
Nov, 1996
ISSN: 1048-535X LANGUAGE: English RECORD TYPE: Fulltext; Abstract
WORD COUNT: 1423 LINE COUNT: 00124

ABSTRACT: Enterprise-wide risk management (ERM) systems can help banks and financial institutions minimize their risk exposure. ERM systems deal with different types of risks such as market, credit and operational that result from a variety of asset classes such as currencies and interest rates across time zones. Effective ERM systems are those that calculate risk characteristics and monitor information that influence risk.

TEXT:

Bank-wide risk management: Practical implementation

Recent high profile catastrophes in derivatives trading have increased awareness among banks and financial institutions of the need for more effective risk management measures, and prompted fresh calls from regulatory authorities for tighter controls. Studies by regulators and academics have analyzed the problems and recommended solutions. In the main, these have pinpointed the need for greater management understanding of the risks involved and better use of technology. Attention has been focused on the substantial investment in research and development from financial software vendors which has established new methods and techniques for monitoring and managing risk.

In essence, the findings underscore the G-30 report which highlighted different aspects of risk management practices and emphasized the role of systems and technology. It stated that firms, both dealers and end users, should establish management information systems sophisticated enough to measure, manage and report the risks of derivatives in a timely and precise manner.

The Many Dimensions of Risk

In the past two years, the concept of enterprise-wide risk management (ERM) has evolved as banks and financial institutions have realized that risk can be multi-dimensional and should include a wide range of definitions.

Most methodologies appropriate to bank-wide risk management originated in derivatives trading activities. While this is interesting and important, it should be noted that risk management is not a formula. Risk Metrics or any other VaR (value-at-risk) methodology is just one potentially valid tool.

An acceptable definition of bank-wide risk management may be: "The systems and procedures designed to deal with the multiple types of risk, such as market, credit, liquidity, operational and legal, that arise from dealing in different asset classes such as currencies, interest rates, equities, and commodities across different time zones."

As risk may arise from many dimensions, the practicality of risk management requires participants to address the following aspects of risk:

Market Risk: uncertainty of future earnings resulting from changes in market conditions surrounding equities, interest rates, foreign exchange and commodity values.

Credit Risk: potential loss due to the inability of a counterparty to meet its obligations.

Liquidity Risk: market impact of buying and selling a **financial instrument**.

Operational Risk: errors that can be made in processing and settling transactions.

Regulatory: changes to, or enactment of, new legislation.

Human Factor: problems ranging from keyboard errors to major fraud.

With so many potential risks there is no logical denial of the need for effective risk management on a global scale. Moreover, today's specifications for any risk management system should take into consideration that present and future requirements lead to more, not fewer, systems and that the importance of integration will remain paramount. Increased regulation, collateralized derivatives clearing houses and standardization of methods such as value-at-risk offer increasing local and global benefits.

The existence of legacy systems and the importance of their integration into a contemporary ERM systems strategy is also vital. Current G-30 recommendations and mooted Bank for International Settlements, Federal

and other regulatory requirements must be accommodated in order to provide the principal goal of achieving consistent, consolidated risk management across all instruments, books and locations.

In order to effectively address those aspects, an ERM system must calculate the following risk characteristics: greeks (delta, gamma, etc.); VaR; storage of historical information; trading and credit limits; simulation/stress testing; back testing.

It must also track all information affecting risk: market prices; trades (locally and globally).

Addressing today's requirements and those anticipated in the future in an ERM calls for solid building blocks. These should cover a sound technology based on industry standards which will continue to evolve, along with crucial functionality such as risk archives, VaR, simulation, source of change, limits and global trade consolidation.

The risk archive stores primary and derived data for historical analysis. Data is stored at user defined frequencies and used for VaR and source of change analysis (e.g. for computing the variance and covariance matrices) and can be used for proprietary analysis.

Value-at-risk measures the maximum loss in market value of a given position that can be incurred until the market risk exposure is neutralized, with a given confidence interval, assuming markets continue to behave as they have in the past. VaR uses historical information stored in the risk archives to determine empirical correlations both intra- and inter-markets at user specified 'confidence levels' and user specified time horizons for position unwind. Simulation provides advanced techniques like scenario analysis, worst/best case analysis as well as forward simulation, Monte Carlo, historical simulation and stress testing.

The source of change analysis (also known as 'what was?' analysis) provides answers to the crucial question: What are the factors that contributed to gain/loss in a portfolio at any given period? More specifically, the technique provides analysis of the source of P&L movements over an entire portfolio, comparison of actual contributions from different risk sources (e.g. delta, time decay, second order changes, etc.) to expected contributions from theoretical projections. Ideally, this should also include a facility to 'drill down' through various data levels to reveal the actual source of change.

Limits capability is important for both market and credit risk and should be monitored in real time against any risk measurement.

For multi-site operations, global trade consolidation is crucial for assessing the risk from any location or system. Thus, the ERM system must ensure that it tracks all global trades. While it may be required to track partial global risk in a hierarchical structure, it also has to track risk as it changes and that "batch" risk isn't typically enough and real-time approach is more practical in today's dynamic market. Global trade consolidation also means that tracking of risk in multiple disparate locations includes multiple applications and multiple platforms and protocols as well as multiple structures.

Bringing Everything Together

Implementation of any effective ERM system will almost certainly involve a complicated configuration of numerous legacy systems and procedures spanning front, middle, back office and accounting practices. Clearly, consolidating trades and positions from many different applications raises serious issues regarding the quality of the data and the actual ability to technically transfer the data from legacy systems to the central data warehouse for risk management purposes.

Further analysis of the effective ERM should, therefore, cover the following aspects:

Definitions: the details on how risk is measured and tracked by the bank.

Transport: an interactive layer which takes data from source systems and makes it available for analysis.

Normalization: a methodology for making risk data similar so that it can be aggregated and analyzed.

Data Warehouse: input and output of risk analysis.

Analytics Package: tools for analyzing the different risk types.

Tracking: tools for comparing business activities to management objectives and other guidelines (e.g. limits).

Reporting: tools for communicating the results of analysis back to decision makers.

If the ERM system is to achieve its key goals for satisfying managers and the regulators, it needs to be based on enduring, open systems technology with client/server architecture, an ultra-friendly graphical user interface and controlled access to structures and models.

The technical and financial requirements for effective risk management can be met. Windows NT technology, along with SQL databases and efficient use of object linking and embedding and **object oriented** technology are key to the success of the implementation. The combination delivers efficient infrastructure for global trade consolidation, data modeling, interfaces to legacy systems and powerful analytical analysis.

A practical approach for risk management will focus on a 'brain' component that will interface with trading and back office systems as well as market data providers. The 'brain' (also known as a risk executive) knows how to process events (e.g. execution of new trades, amendment of existing trade, change in market rates/volatility etc.) and how to transport the data to the risk warehouse.

Once a comprehensive risk warehouse is established, analytical packages (risk workstation) can be applied for VaR, back testing, simulation, credit risk analysis and other advance techniques for effective risk management.

Hagay Shefi is managing director for North and South America at SunGard Capital Markets.

SPECIAL FEATURES: illustration; chart
INDUSTRY CODES/NAMES: BANK Banking, Finance and Accounting; INTL Business, International; BUSN Any type of business
DESCRIPTORS: Risk management--Technique; Derivatives (Financial instruments)--Management; Banking industry--Management
PRODUCT/INDUSTRY NAMES: 9915300 (Asset & Risk Management); 6010000 (Banking Institutions)
SIC CODES: 6000 DEPOSITORY INSTITUTIONS
FILE SEGMENT: TI File 148

4/9/26 (Item 3 from file: 148)
DIALOG(R)File 148:Gale Group Trade & Industry DB
(c)2000 The Gale Group. All rts. reserv.

08492088 SUPPLIER NUMBER: 18038391 (THIS IS THE FULL TEXT)
Objects take off; PepsiCo to install what could be a huge **object-oriented** app to increase its market agility. (**object-oriented** applications) (includes related article on user benefits of object technology) (Technology Information)(Cover Story)
Bartholomew, Doug
InformationWeek, n568, p14(2)
Feb 26, 1996
DOCUMENT TYPE: Cover Story ISSN: 8750-6874 LANGUAGE: English
RECORD TYPE: Fulltext; Abstract
WORD COUNT: 1249 LINE COUNT: 00107

ABSTRACT: Marcam's delivery of its Protean production and inventory software to PepsiCo may be the largest implementation of packaged **object-oriented** applications to date. The **object-oriented** applications offer Pepsi ease of use and flexibility. In-house corporate developers, particularly in financial services, have used and reused objects when building custom applications. Ready-made **object-oriented** applications, which require little development expertise, expand upon this base by meeting the changing demands of business needs without any changes to underlying code. Despite recent attention to object technology, technology users remain slow to utilize

object software packages because the technology is relatively untested and young. Others may be reluctant because they do not foresee immediate pay-offs.

TEXT:

In China, 1996 is the year of the rat. But in the world of enterprise applications, 1996 is shaping up as the Year of the Object. In what eventually could turn into the largest implementation yet in the nascent market for packaged **object-oriented** applications, Marcam Corp. expects to deliver a version of its Protean production and inventory software on March 5 to PepsiCo Inc.'s Pepsi-Cola North America division.

Pepsi is one of several big companies seeking flexibility and ease of use by buying **object-oriented** enterprise applications. "Everybody is suddenly discovering how hot objects are going to be this year," says Bruce Richardson, VP for research at Advanced Manufacturing Research, a technology research firm in Boston.

For some time, in-house corporate developers, especially in industries such as financial services, have used objects--reusable chunks of software code--to save time in building custom applications. While that practice continues to grow, what's different about the latest trend is that companies are buying ready-made **object-oriented** applications, which require little or no development expertise. The payoff for companies comes not in developing the original software, but in modifying the application to adapt to a changing business without having to touch the underlying code. The desired result: a nimbler company.

"Adaptability is the key," says Ray Sasso, corporate chief information officer at J.R. Simplot Co., a $2.8 billion food processor in Boise, Idaho.

Information systems managers should expect to see an increasing number of software vendors offering object-based applications. "Objects are the future of the business because they require very little custom programming," says Doug Magill, a consultant and former CIO at Nestl -Stouffer Co. in Solon, Ohio.

In addition to Marcam, a number of vendors, most notably System Software Associates in Chicago and Systems & Computer Technology Corp. in Malvern, Pa., already have customers using their enterprise-level **object-oriented** packaged software. Dun & Bradstreet Software Inc. has an **object-oriented** workflow capability in its enterprise application suite. Other vendors, including QAD Inc. in Carpinteria, Calif., and Sherpa Corp. in San Jose, Calif., will soon release object-based applications.

If all goes well with a $3 million project to test Protean at one of Pepsi's bottling plants, the company intends to install the software to link all its North American bottling and warehousing activities--65 bottling plants and 240 distribution centers--in a common system for managing production and inventory of the division, which has $6 billion a year in sales. The initial project, scheduled to be up and running by June 1, will include some 200 users with Microsoft Windows PCs. That would expand to include thousands of users throughout the company's supply and distribution network.

The flexibility offered by **object-oriented** software was apparently a key factor in Pepsi's decision to install Protean. "Its business changes rapidly, and the company felt that the adaptability and flexibility inherent with our object architecture would make it easier for them to reflect these changes in the system," says Mark Arsenault, an account manager at Marcam in Newton, Mass.

"We see what we are doing with Protean as having a potential for achieving competitive advantage," says Ken Gerhardt, director of application development at Pepsi-Cola North America.

To win the Pepsi contract, Marcam beat out Dun & Bradstreet, which already had a foothold because Pepsi uses D&B's mainframe-based financial software. For Pepsi, installing Protean is a logical extension of the company's substantial in-house **object-oriented** application development effort using the PowerBuilder development package from Sybase Inc.'s Powersoft unit, according to Marcam, which says some of Pepsi's internally developed objects can be integrated with Protean.

Pepsi isn't alo[redacted]in its desire to use objects t[redacted]ct quickly. Marcam, which has sold Protean since late 1994, recently sign[redacted]deals with such major companies as 3M Co. and Sun Chemical Corp., as well as with Simplot.

With any fast-changing business, no application will last long. "In that context, you can't have an investment in old, inflexible applications that are difficult to maintain," says Steve McClure, director of object tools at International Data Corp., a Framingham, Mass., market research firm.

For example, in financial services, where **object-oriented** development is common, "every day someone's inventing a new **financial instrument**," McClure says. "Those companies can't wait two years to respond."

But fast reaction times aren't just for financial firms any more. Simplot, the Idaho food processor, began using Protean at one plant last fall and at a second plant in January. "The object orientation provides us with so much customizability," Sasso says. "It makes it easy to adapt the system to business changes very readily."

Sasso also says it was easy to train workers to use the system to record shipments, update inventory, schedule production, and track orders. Roughly 80% of the workers had never touched a computer before. Simplot figures it will spend $15 million to $25 million to roll out the software to its entire 38-plant operation over the next four years.

Ramsey Sias Co., a Brecksville, Ohio, maker of fruit fillings for companies such as Dannon and Haagen Dazs, began using Systems & Computer Technology's **object-oriented** Adage software in January to manage purchasing and accounts payable. "The company's users can see their business process expressed as objects on the screen, and they are able to grasp the different steps involved pretty quickly," says Magill, a consultant for Ramsey's object software project.

Despite the burst of object activity, observers say technology users have been slow to embrace the software packages because the technology is so young and relatively untested on a scale the size of Pepsi's plans. Others may be hesitant because they perceive no immediate payoff. "It's still at the leap-of-faith stage," says Richardson, the manufacturing technology analyst.

Maybe, but a growing number of big companies are taking that leap.

Related Article: Objects: A New Twist

Pepsi's selection of an **object-oriented** application is a new twist on the "objects are good for you" story. We've heard object technology touted for years as a tool for making developers nimble in building applications. But now we're finding that those same technology characteristics--encapsulation, inheritance, and polymorphism--can deliver benefits to users of object-based applications as well.

Off-the-shelf software built using object technology can offer flexibility to a business. Object-based applications are easily changed because they're built using reusable, modular components. A good application design passes that adaptability on to the user--letting an organization assemble components to match applications to its business model.

Inheritance lets the information systems group alter a base class, such as monthly statements, to reflect, for example, a new discount policy. Because the functions of that object are hidden-encapsulated--users don't need to know how the object works, only how to use a high-level scripting language to change its behavior.

Polymorphism, in which different objects can plug into the same interface, allows the output of a report to be sent to a local printer, faxed to a branch office, or E-mailed to an executive on the road without any changes to the source code.

This level of customization could be provided using traditional development methods, but it would take longer to build-and in a competitive market, time is of the essence.

Julie Anderson is technical director of InformationWeek's OpenLabs.

SPECIAL FEATURES: illustration; other

COMPANY NAMES: Pep[redacted]o Inc.--Data processing
INDUSTRY CODES/NAMES: CMPT Computers and Office A[redacted]omation
DESCRIPTORS: **Object-oriented** programming--Usage; Computer software industry--Innovations
PRODUCT/INDUSTRY NAMES: 7372420 (Database Mgmt Software Pkgs (Micro))
SIC CODES: 7372 Prepackaged software
TICKER SYMBOLS: PEP
FILE SEGMENT: CD File 275

4/9/27 (Item 4 from file: 148)
DIALOG(R)File 148:Gale Group Trade & Industry DB

08111242 SUPPLIER NUMBER: 17258977 (THIS IS THE FULL TEXT)
The open view. (Unix-based transaction monitors)
Richman, Dan
InformationWeek, n543, p45(4)
Sep 4, 1995
ISSN: 8750-6874 LANGUAGE: English RECORD TYPE: Fulltext; Abstract
WORD COUNT: 1749 LINE COUNT: 00146

ABSTRACT: Enhancements in three-tiered client-server architecture, online transaction processing rates and heterogeneous database synchronization have closed the technological gap between Unix-based transaction monitors and mainframe monitors. Unix-based transaction monitors are less costly to develop, operate and install, and market revenue is estimated to increase to $537 million in 1998, a 500% raise from 1994. Novell Inc's Tuxedo transaction monitor controlled 32% of the 1994 market. However, the transaction monitor still slightly trails its mainframe competition in terms of application efficiency, batch function performance and the small number of users it can handle. However, the substantial revenue-savings potential of transaction monitors may compensate for what they lack in power.

TEXT:

Unix transaction monitors help businesses synchronize transactions, implement three-tier client-server architectures, and boost throughput. They could even displace mainframe monitors.

How does a bank ensure that a customer's transfer order actually credits the right account while debiting another? How does an engineering firm use a groupware application to pass off a computer-assisted design for an engine bolt to a supervisor for examination and approval before that design is shipped off to a factory? u The answer is the transaction monitor, also known as a transaction manager. Once found only on mainframes, Unix transaction monitors synchronize transactions between heterogeneous databases, implement three-tiered client-server architectures, boost the throughput of online transaction processing, and increase the number of users that a system can support.

Some users and analysts deride Unix transaction monitors as too complex and immature for widespread use. But the technology-really a form of middleware-plays an increasingly important role in corporate client-server environments.

Transaction monitors help guarantee that data-processing transactions succeed. Or, if the process fails, they make sure the failure won't damage the integrity of the data. Monitors also allow load balancing, or the spreading of applications among different machines for maximum efficiency. They also offer various remote procedure calls that link the layers in a three-tiered client-server set-up, helping a business choose the right tier for the right piece of the system.

Helping Hands

There are several reasons for the growing popularity of Unix transaction monitors. The technology tends to be less expensive than their mainframe transaction-oriented database counterparts, such as IBM CICS, IBM IMS, and Computer Associates CA-IDMS. The Unix version also requires fewer administrators, is easier to program, and tends to use cheaper hardware.

"We're getting [redacted]formance that's just as good a[redacted]e're used to on the mainframe," says Jim [redacted]tman, VP of system architectu[redacted] Cincinnati Bell Information Systems Inc. (CBIS) in Cincinnati. But Holtman and other users have words of caution. They say tools to manage Unix transaction monitors are uncommon or inefficient, can't handle many users, and don't perform batch functions as well as mainframe monitors. They're even worried about the maturity of related functions in the Unix world such as backup and recovery.

That's why CBIS, a subsidiary of Cincinnati Bell Inc., uses both Unix and legacy transaction monitors. Its Unix system is based on the market-leading transaction monitor, Tuxedo, from Novell in Provo, Utah. But the majority of processing at CBIS is still based on the IBM CICS database and transaction monitoring system.

Over the past five years, CBIS has processed more of its bills and customer service calls with an internally developed system called Precedence 2000, built around Tuxedo. With Precedence, CBIS performs billing and online customer services for cellular phone companies. Unlike the mainframe systems at CBIS, Precedence 2000 runs in a distributed fashion, spreading transactions over multiple servers to increase computing efficiency. That's a key difference compared to mainframe monitors. Given the nature of Unix systems, the management of distributed processing is vital.

Precedence 2000 helps 80 service representatives, using PCs running Microsoft Windows, respond to nearly 200,000 customer calls daily. Holtman says the Tuxedo-based system works so well that all new customers are put into it, leaving only long-standing customers in CICS systems.The company prefers the C and C++ development languages, which work well with Tuxedo for extensible, **object-oriented** development. CBIS likes Tuxedo so much that eventually it may try to move all its operations off the mainframe. But Holtman acknowledges there are drawbacks. "The critics have some good points to make," he says. "In some respects, Unix transaction managers aren't as capable as mainframes. If someone would invent a way to accurately translate Cobol automatically into C, it would sure help us out."

Though Precedence processes 200,000 transactions each day, the mainframe can handle 10 times that volume. CBIS believes the mainframe far outstrips any Unix system at inputting huge volumes of data on tape, performing backup and recovery, and executing major batch jobs and sorts.

Leap Of Faith

More businesses are experimenting-or like CBIS, running strategic applications-with Unix transaction monitors. Sales of the monitors will hit $537 million by 1998, a fivefold increase over 1994's sales of $109 million, according to projections from the Standish Group International, a consulting firm in Dennis, Mass. (By contrast, the total market for non-Unix transaction monitors was more than nine times greater last year at nearly $1 billion.)

One user, GE Capital Mortgage Corp., a seller of mortgage insurance in Raleigh, N.C., uses the technology to help link 200 users in 26 branch offices around the country to a Sybase database in Tennessee. The goal is to record all sales in real time. The company uses Encina, a transaction monitor from Transarc in Pittsburgh, to ensure the integrity of the transactions. The information is stored on a mainframe but soon will be moved to Sybase, and possibly Oracle, under some combination of Unix and OS/2.

"We feel confident enough that we're going to use the mainframe only for storage and take away any transaction processing role," says Stan Patterson, a senior technical analyst at GE Capital. "It's particularly valuable for the heterogeneity it allows."

Ed Wehner, manager of business information services at Memc Inc., a silicon wafer producer in St. Peters, Mo., rebuts the criticism that a Unix transaction monitor can't be used effectively for batch processing. Using CICS/6000, an AIX and HP-UX version of IBM's CICS, Memc runs batch transactions of customer order, scheduling, and labeling processes even while the system is operating online. When Memc used CICS on the mainframe, the company couldn't run batch and online queries at the same time.

"There's nothing we can't do better under Unix than on the mainframe," Wehner adds.

Managing Growth

Using Unix transaction monitors cuts personnel costs, too. While Wehner used six or seven people to run his mainframe operations, today the same monitoring tasks can be handled by a single analyst.

One of the reasons for the proliferation of Unix transaction monitors over the past five years is that more companies need the technology to help them manage three-tiered client-server systems. In three-tiered systems, clients (containing the interface) connect to applications (containing the program logic), which in turn interact with databases and other computing resources.

The number of three-tier systems will grow by nearly 75% between now and 1997, predicts Strategic Focus, a consulting firm in Milpitas, Calif. Three-tiered systems make up only 5% of the total number of client-server applications. By 1997, this architecture will make up nearly 20% of the total, say Strategic Focus analysts.

Essential to the successful implementation of three-tiered architectures is some way of managing the interactions among the layers. Transaction monitors help companies accomplish this, says Ivan Ruzic, Novell's director of marketing for Tuxedo. The alternative, custom-written middleware, is tedious and difficult to create.

One East Coast financial institution implemented a three-tier setup two years ago using Tuxedo. PCs running Windows access applications Pyramid servers, which in turn access data on IBM mainframes. "If we didn't have Tuxedo, there's no way we could have created this system, which is helping handle essential functions like cost-based accounting, **financial-instrument** reporting, and commission calculation," says a senior technical staffer at the company.

The major transaction monitors support multiple databases. That makes transaction monitors popular not only with companies that operate several database platforms, but with commercial software developers as well. Among the most outspoken is Larry Tanning, president of Tanning Technology Corp. in Denver. "Anyone saying Unix transaction managers aren't ready for prime time would sound like an idiot to the 4,000 sites where we've installed our software based on them," he says.

Tanning's clients agree. Gordon Divitt, president of Fund Serv Inc., a mutual-fund network in Toronto, says he's completely satisfied with Transaction Forwarding System, a Canada-wide, three-tier mutual-fund purchasing application that Tanning developed with two partners. "I got what I wanted," says Divitt. "Development was quick. It operates efficiently. It's stable and easy to extend."

But Tanning agrees there are still important weaknesses in the technology. Unix system- and network-administration tools lag those for mainframes.

Indeed, analysts like Rich Finkelstein, president of Performance Computing in Chicago, say Unix transaction monitors are still far too difficult to install, administer, and use as a base for writing programs. "We need simplification," he says. Roy Schulte, an analyst with Gartner Group Inc., an IT advisory firm in Stamford, Conn., says monitoring tools for Unix transaction monitors aren't up to the level of mainframe products. For some compute-intensive processes, Schulte adds, companies worry that Unix tape handling isn't good enough, and that batch processing is still not up to mainframe speeds.

Future Remedies

Vendors of Unix transaction monitors are working to remedy the limitations. Sometime next year, Tuxedo will integrate more closely with Novell Directory Services, according to Tuxedo marketing director Ruzic. That will let an administrator control Tuxedo-based applications from a single monitor, along with NetWare LANs and devices that run under Novell's Embedded Systems Technology. When Novell starts supporting a Simple Network Management Protocol agent for Tuxedo, administrators will be able to control Tuxedo using HP's Open View, CA's Unicenter, or similar products.

Top End, Tuxedo's rival from AT&T GIS, already can be managed from most major systems-administration tools, says business unit manager Randy

Smerik in San Diego. Now, the company will focus on porting the product to Windows NT.

Despite shortfalls, companies are lured by the technology. "There used to be big gaps in functionality compared with mainframe monitors|," says Mike Prince, director of MIS for Burlington Coat Factory Warehouse in Burlington, N.J. "Today, it's down to the icing on the cake instead of missing an oven to bake the cake."

That oven is getting fancier by the day.

SPECIAL FEATURES: illustration; photograph; table; graph
INDUSTRY CODES/NAMES: CMPT Computers and Office Automation
DESCRIPTORS: Data base management systems--Usage; Computer software industry--Marketing
PRODUCT/INDUSTRY NAMES: 7372420 (Database Mgmt Software Pkgs (Micro))
SIC CODES: 7372 Prepackaged software
FILE SEGMENT: CD File 275

4/9/28 (Item 5 from file: 148)
DIALOG(R)File 148:Gale Group Trade & Industry DB

07668684 SUPPLIER NUMBER: 16368109 (THIS IS THE FULL TEXT)
Big banks license Cats' design software. (Cats Software Inc., Ficad financial software, Bankers Trust New York Corp., Tokai Bank Europe, Mitsubishi Finance International) (Brief Article)
Epper, Karen; Marjanovic, Steven; Barthel, Matt
American Banker, v160, n19, p15(1)
Jan 30, 1995
DOCUMENT TYPE: Brief Article ISSN: 0002-7561 LANGUAGE: ENGLISH
RECORD TYPE: FULLTEXT
WORD COUNT: 105 LINE COUNT: 00009

TEXT:

PALO ALTO, Calif. - Cats Software Inc. announced that several large banking companies have licensed its computer-aided-design financial software, known as Ficad.

The institutions include Bankers Trust New York Corp., Tokai Bank Europe, and Mitsubishi Finance International.

Based on an **object-oriented** approach to programming, Ficad enables traders, risk managers, and product developers to create in a few days applications to monitor any **financial instrument**.

"Computer-aided design tools have been successfully applied in every other major industry over the past decade," said Rod. A Beckstrom, chief executive of Cats. This software "brings the power of [computer-aided design] to finance.

COMPANY NAMES: C.ATS Software Inc.--Licenses; Bankers Trust New York Corp.--Licenses; Tokai Bank Europe--Licenses; Mitsubishi Finance International--Licenses
INDUSTRY CODES/NAMES: BANK Banking, Finance and Accounting
DESCRIPTORS: Computer software industry--Licenses; Banking industry--Licenses
PRODUCT/INDUSTRY NAMES: 7372201 (Financial & Acctg Software Pkgs); 6010000 (Banking Institutions)
SIC CODES: 7372 Prepackaged software; 6000 DEPOSITORY INSTITUTIONS
TICKER SYMBOLS: BT
TRADE NAMES: Ficad (Banking/finance/investment software)--Licenses
FILE SEGMENT: TI File 148

4/9/29 (Item 6 from file: 148)
DIALOG(R)File 148:Gale Group Trade & Industry DB

07666897 SUPPLIER NUMBER: 16497681 (THIS IS THE FULL TEXT)
The case for expressive systems.
Pawson, Richard; Bravard, Jean-Louis; Cameron, Lorette
Sloan Management Review, v36, n2, p41(8)
Wntr, 1995
ISSN: 0019-848X LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 6020 LINE COUNT: 00493

ABSTRACT: A new kind of information system is emerging that will reduce the time to market, help tailor products and services to customers' needs, and make processes more responsive to unexpected events. Expressive systems allow users to adapt quickly and easily to exceptions from standard operating procedure. The authors describe how expressive systems work and suggest ways of modifying the roles and structure of IS departments to implement the new technology. (Reprinted by permission of the publisher.)

TEXT:

On the derivatives trading floor at J.P. Morgan in New York, there is a new information system called Kapital. The trading environment supported by this system is a demanding one: the traders who use it are at the cutting edge of creativity in the financial markets. In addition to trading a wide variety of instruments, they are continually inventing new instruments by combining parts in new ways, "bundling," or fashioning an entirely new set of custom terms and conditions. Conventional applications development approaches do not easily support the degree of systems flexibility required by such an environment.

Kapital employs some of the most sophisticated technology currently fashionable in the world of information systems today: it is based on a distributed client/server architecture, borrows techniques from the world of artificial intelligence, and is one of the purest implementations of the concept of **object-oriented** software in the business community.
Using Kapital, traders can choose the user interface that suits them, from simple business forms to graphical representations. They can perform powerful financial analytics using complex mathematical models. Kapital accommodates real-time data feeds giving current market information and performs sophisticated portfolio analysis against a variety of market assumptions.

However, what really distinguishes Kapital from other information systems is not the technology, but the fact that it does not attempt to fulfill a specified set of user requirements - at least in the conventional sense. Rather, Kapital attempts to model the very "language" of J.P. Morgan's trading business - not only the vocabulary, but also the grammar and, arguably, the style. Kapital implements that language in software, so that the traders can directly express their ideas for new "exotic" tradable instruments. One objective for the system was that, as fast as traders could conceive a valid opportunity for a new kind of **financial instrument**, he or she should be able to directly interact with the system to create that instrument, simulate its performance in terms of risk and profitability, and if satisfied, price and trade it immediately. To do this required that the software present atomic-level financial components and business rules to the traders, along with the capability to manipulate and extend them in new ways.

Kapital is not a programming language in the conventional sense; it bears no resemblance to the so-called "fourth generation" programming languages on the market today, nor even the graphical programming tools now gaining popularity. Its basic constructs are not simply high-level representations of the computers resources (although these are provided where useful), but the natural constructs and components of the trader's world: cash flows, interest rate scenarios, risk profiles, and so forth.

While giving users direct manipulation of the systems capabilities, Kapital does not eliminate the need for professional programming. A high-caliber team needs to maintain and continuously enhance the business language and its supporting technologies. Furthermore, traders may call on

professional programmers to assist them in implementing a more difficult idea, refining a prototype they have developed, or writing a new component from scratch. However, the response time for such requests is measured in minutes and hours, not weeks and months. In part, this is because Kapital provides high-level business capabilities and low-level technical components in one integrated environment. The developers do not have to translate a requirement from a business domain, as with conventional programming, into a different technical medium.

Kapital is an example of a new kind of information system that is beginning to emerge in business, which we have dubbed "expressive systems." Conventional systems support only the standard business operating procedures for which they were designed.[1] Expressive systems are designed to support exceptions from standard operating procedures, empowered actions, new product ideas, services tailored to individual customer's needs, ad hoc or even temporary changes in organizational structure - none of which, by definition, can be specified up front. These changes can be implemented in a time frame appropriate to the business need - in some cases, in seconds or minutes, by the users themselves; in more complex cases, in hours or days, with the help of professional programmers; but never in terms of weeks or months.

Expressive systems not only make it easy to implement these changes, they encourage business managers or empowered workers to explore more possibilities for change and thus improve business performance. Using an expressive system within any of these contexts feels as natural as using an electronic spreadsheet program to develop a financial model and then explore "what-if" scenarios.[2] But expressive systems are not mere simulation tools. They permit the user to execute the action through the same system, whether that result is a new **financial instrument** to trade, a new way of routing documents through an administrative function, or a reallocation of work orders between manufacturing plants.

In this paper, we show, with reference to both examples and new theory, that expressive systems will in time become the dominant paradigm for business computing. Implementing expressive systems, however, will require organizations to address new technologies and redevelop many of their existing systems. Furthermore, the implementation and support of expressive systems will require a substantial modification to the roles and structure of the information systems department.

Roles for Expressive Systems

We have identified three specific roles that expressive systems will play in business:

* Reducing the time to market for introducing new products.
* Facilitating the tailoring of products and services to individual customer's needs.
* Making operational processes more responsive to unforeseen events.

(Interestingly, these three roles correspond to Treacy and Wiersema's three dimensions of market leadership: product leadership, customer intimacy, and operational excellence.[3] This suggests that expressive systems have a role in all organizations that seek to lead their markets.)

Reducing Time to Market

In many industries, from automobiles to pharmaceuticals, reducing the time to bring new concepts to market is critical. Computer-aided design systems, arguably an early form of the expressive system concept, have encouraged users to explore more design alternatives and better understand the consequences of their actions. The new Boeing 777 aircraft was completely designed on computers using a package called Catia. For the first time in modern aircraft design, it was not necessary to build an evolving full-scale prototype to find out if the pipework and other systems could be routed through the narrow confines of the airframe. The computer simulation provided that intelligence - the first 777 was built to fly.

The advance of new prototyping technologies, such as stereo lithography, which can create a plastic three-dimensional form from a computer model in seconds, and robotic assembly means that the users of such systems can express their ideas directly into physical form. However, the greater the information content of products and services, the stronger the potential for expressive systems to reduce the product development

time.

In a limited range of cases, expressive systems can potentially eliminate the product development process altogether. Arguably this is the case with J.P. Morgan's Kapital system. The ability to create new financial instruments "on the fly" changes the trading paradigm from looking for opportunities to trade each of a set of preexisting instruments to creating the instrument needed to take advantage of each opportunity that arises.

While the financial trading environment is undoubtedly a rarefied one, it is worth noting that there is a clear pattern of innovations transferring from this environment to "ordinary" business.[4] Telecommunications companies and energy utilities, for example, are starting to deploy systems of similar sophistication in order to introduce new forms of billing and new value-added services, in response to rapidly changing market conditions.

Tailoring Products and Services to Customer's Needs

Many organizations recognize that to retain their most valued customers, they must increasingly respond to an individual customer's needs. The difficulty lies in being able to do this with something approaching the economy of standardized services; information systems are clearly the key to this objective. Banks and other financial services have for years been developing "parameter-driven" information systems that would allow them to vary the parameters of a financial product (the interest rate, term, and any discounts) without the need to write new program code. In reality, however, the degree of customization this approach offers is still small, and the professional systems effort needed makes it uneconomical for application to individual customers. One organization that is seeking to break this limitation is Britain's National Westminster Bank (Nat West).

As part of a series of initiatives to define the future of retail banking, the IT strategy department at Nat West developed sophisticated PC-based software that would permit not only the parameters, but the very operating structure, of a bank account to be tailored. Suppose, for example, that a high net-worth customer preferred to have her checkbooks sent, not to her home, but to the nearest branch of the bank - with an advisory letter sent to her home. This apparently simple requirement would be beyond the scope of most parameter-driven systems and would require expensive manual intervention. Nat West's prototype system permits this scenario simply to be drawn out graphically on the screen, creating the necessary supporting systems automatically.

Several things about Nat West's prototype system stand out:

* The software was designed to be used by an IT-literate bank manager or, more probably, by a systems professional sitting next to the bank manager. But, either way, the new type of account is created in real time, typically in response to a customer request.

* The system is graphical, with icons representing all the business constructs that a bank manager would expect to deal with - customer, interest rate calculator, statement, checkbook, and so forth.

* The system does not present the user with a series of predefined options. Rather, it feels like a well-designed child's construction kit, with which the user can quickly explore and implement almost any idea.

* The user can see the profitability, or otherwise, of the financial product. (This can even be done on a separate screen, allowing the basic product to be designed in front of the customer.)

* Built-in constraints prevent the user from building illegal or nonsensical financial products.

It may be several years before this scenario pervades the banks branch network, but Nat West is committed to implementing its system and probably has at least a couple of years' lead on its competitors. The difficult part is not designing the graphical front end, but re-designing the deep structure of the core information systems to allow them to be manipulated this way.

Expressive systems therefore perform two important functions in product or service customization. The first is to permit the customization to take place in "real time" at the point of customer contact, rather than later in a back office. The second is to permit the user to explore and

understand the consequences (here, in terms of profitability) of the proposed approach or action. Without this capability there can be no substance to the concept of empowered actions. This theme carries over into the third function.

Responding to Unforeseen Events

In the context of improving operational performance, expressive systems take on two roles: the first is to optimize the refinement of resource use; the second is to facilitate the handling of unexpected events and potentially chaotic disruption.

Manufacturing, logistics, and other key operational functions have been subject to intensive study and refinement for many years. Both quality and efficiency have been driven up, waste reduced, and nonproductive costs, such as stock, virtually eliminated. The scope for further refinement is narrowing rapidly. But the refinement has brought a new problem: greater potential for chaotic disruption (we use chaotic in the mathematical sense).

American Airlines, whose operating procedures and information systems are second to none, has recognized this. Its systems operational control is the business unit charged with executing the flight schedule and marshaling the many different resources on which it depends. Any flight may draw its plane, flight crew, and cabin crew from three different incoming flights, while baggage handling, gate space, catering, cleaning, and other ground-based resources must also be coordinated. With this level of dependency, unforeseen events, from unscheduled maintenance to adverse weather (not to mention presidential haircuts!), have the potential for chaos. Currently, the antidote means preserving the structure of the dependencies at all cost - even if this means delaying whole complexes of flights. No one really knows the true cost of these off-schedule operations because of the difficulty of separating the complex costs from routine operations and estimating the impact on customer loyalty, but they are believed to be enormous.

Reducing this cost cannot, by definition, be achieved in the same manner as previous operational refinement. As part of a long-term program to apply the power of IT to this thorny problem, American Airlines has developed new systems specifically to support its flight$dispatchers, the individuals who manage a flight from the ground, including all resources and any route changes. Previously, flight dispatchers had to access information systems through the same kind of transaction-oriented interface as the reservation systems. The new system not only provides more direct manipulation of the system, but also helps the dispatchers to explore the consequences of each unscheduled event and each possible response. One feature of the user interface, for example, is a time line that graphically portrays the prior events on which a particular flight depends, future flights that in some way depend on it, and their current status. Dispatchers can instantly see the consequence of shifting the proposed take-off time, in terms of disruption to the schedule, to staff and internal resources, and, of course, to passengers. No airline currently has an effective model of the financial costs of such disruption, or of the impact on future revenue from customer dissatisfaction, but the American Airlines system is a significant step in that direction. One way of looking at this type of expressive system is that it takes the conventional concepts of operations research (including, for example, PERT networks) and makes them an integral part of real-time operational systems.

A second example is Black & Decker (B&D), whose complex manufacturing operations have been based for twenty years on MRP II, the standard for manufacturing resources planning. An increasing proportion of B&D's sales is coming from large and streamlined retail operations, such as Home Depot, that may place orders on the basis of "deliver within seven days or the order is canceled, and we devote the shelfspace to competitive products." With some retail stores four days away by road, this gives B&D just three days to respond. But the MRP II systems require so much setting up and processing time on mainframe computers that they can be run only every seven days.

Moreover, the MRP II algorithm works in one direction only: it converts a master build schedule into a materials requirements plan and

thence into a capacity plan. If there is an unexpected change to the availability of manufacturing capacity, or to the availability of materials, the MRP II system cannot identify the consequences or advise alternative actions.

Against this backdrop, B&D formed a new team for advanced manufacturing technology with the goal of designing the manufacturing control system of the future. That system, built with the help of a small but innovative software vendor called Intellection, is now in operation in several of its plants. B&D believes that it now has an operational flexibility that not even the Japanese can match. It allows the company to consider the consequences of any event and dynamically explore alternative ways to meet the same requirements. In the not-too-distant future, the system will be accessible from every machine cell in the plant, enabling individual machine operators to identify and understand the consequence of, say, shutting down the machine for an hour's preventative maintenance and finding alternative ways to get the parts made in the meantime.

Thus a key role for expressive systems in high-performance operations is to reduce the potentially chaotic effect of disruptive events. Henry Mintzberg wrote, "When the planners run around like Chicken Little crying, 'The environment is turbulent! The environment is turbulent!,' what they really mean is that something has happened which was not anticipated by their inflexible systems."[5]

Expressive Systems Contrasted with Other Systems

There are other kinds of information systems, such as decision support systems, and other new approaches to systems development, such as rapid application development, that are seeking to address some of these same business goals.[6] However, there are also some clear distinctions, and it is these distinctions, we believe, that make the expressive systems approach more effective in meeting those goals.

Decision support systems, or executive information systems, have made it possible for users to express their information requirements directly, and their ease of use has encouraged managers both to analyze past performance in greater depth and to simulate better the possible consequences of proposed actions.[7] However, the functionality of executive information systems is typically limited to obtaining and analyzing information - they are not a medium through which actions can be executed, in contrast to each of the examples discussed above.

Furthermore, creating a decision support system is usually a matter of grafting new software on to the front end of existing transactional systems. The front end, whether an off-the-shelf package or specifically designed for its purpose, shields the user from the technical details of accessing the underlying systems, provides powerful data manipulation tools, and typically wraps the whole in a nice graphical interface. Newer generations of such packages permit the user to change data stored in the underlying systems, for example, to change a customers address, and perhaps to invoke standard procedures, such as "issue a statement" without leaving the graphical environment. But the only actions or functions available to the user are the fixed set of transactions that the underlying system was designed to support.

If expressive systems are to support actions not previously specified, the user needs access not only to predefined transactions, but to the building blocks from which new kinds of transactions can be constructed. Most of today's core transactional and other "mission critical" information systems were not written with this objective in mind and do not support access at this component level. Some very modern, large systems include application programming interfaces (APIs), which provide better access to the underlying functionality, but these are intended for professional programmers who will be constructing substantial new software applications. Creating the component level access required to implement the expressive systems concept typically requires a complete rewrite of existing core systems.

Within the IS community, the biggest competitor to the concept of expressive systems is probably rapid application development (RAD)[8] - the new techniques for dramatically reducing the lead time to develop new systems. Some instances of RAD deploy similar technology to that in

expressive systems (including client/server and object orientation). Some use conventional systems technology and programming languages but deploy different management techniques such as intensive workshops involving users and developers (sometimes called joint application development, or JAD), and time-boxed deliverables.

The significant difference between the RAD approach and the expressive systems approach lies not in the technology or the actors, but in the intent of the process. JAD and RAD are fundamentally techniques for getting better agreement and ownership of the required specification, either through intensive user/developer workshops at the start of the process, or through an iterative process of delivering crude prototypes of systems and refining the specification based on user feedback from those prototypes, toward a stable end point.

In the expressive systems approach, the iteration is primarily away from a stable start point. The basic components and capabilities of the system provide the start point, but as individual business units or individual users use them, they will move away from the start point as their needs change.

Implementing Expressive Systems

Expressive systems therefore change the concept of an application. Today, the term "application" refers to a collection of programming code and data that together meet a neatly circumscribed and well-defined set of business requirements, such as an order-processing system or a credit management system. Applications account for the greater part of the budget, manpower, and management attention of most IS departments. The applications are supported by a common technical infrastructure, whose costs and management are shared, but these are smaller by proportion. Implementation of an expressive system requires an inversion of this balance, with a much thinner applications layer and a much thicker (or richer) shared infrastructure, which comprises not only technical components but also business constructs.(9)

If they are thin enough, applications can be thought of as a wiring layer. We find that this notion has particular appeal to systems professionals old enough to remember analog computers, in which applications were constructed by wiring together standard components on a plug-and-socket panel. Moreover, some of the PC- and workstation-based software tools most appropriate to building expressive systems (for example, Digitalk Parts, NeXTStep, and IBM's VisualAge) permit software components to be visually wired together on screen.

We could go farther and say that the word "application" changes its sense from a noun to a verb (strictly, the gerund of a verb).[10] Application now refers to the process through which the infrastructure is applied to the needs of a business situation or an individual user, rather than to a piece of software in its own right.

So what does the thicker infrastructure actually contain? The sine qua non for expressive systems is an "uncommitted" software model of the business (or the particular domain of the business that the system is to serve). In microelectronics, an uncommitted array is a silicon chip that is made as a standard component but, in the final stage of manufacture, is committed to a specific customer application, such as the video circuitry for a games machine or the ignition control system of a car. Similarly, an uncommitted software model represents a business in generalized form but can be committed (or recommitted) to a particular product set, market channel structure, or business organization.

For some years, there has been limited implementation of this concept in the form of tailorable software packages, which are now gaining in popularity. Here, the user organization has the ability to choose between a number of options (possibly a very large number), predetermined by the vendors of the package. In a truly uncommitted software model, however, the designers have not attempted to foresee all the possible configurations. It is like the difference between a model car that can be customized with decals and accessories, and a Lego set.

Designers of children's construction sets face a constant trade-off Versatility requires more different kinds of components and lower-level components (individual wheels and bricks). Ease of construction demands

fewer components, and this typically translates to ready-made subassemblies, like a vehicle cab or house roof. Designers can partially overcome this by creating powerful high-level components that take on several different roles. This principle is called "abstraction," and it is the key to building powerful uncommitted software models of the business. The following example illustrates why:

Three years ago, the Bradford & Bingley Building Society (roughly equivalent to a savings bank) replaced most of its systems portfolio with a new system, developed from scratch and based on an uncommitted business model. The old system was proving increasingly costly to maintain and needed replacement anyway. The principal objective, and the reason for the choice of approach, was a system that no longer constrained innovation. The chief executive himself stated that he did not want to be told that he could not implement an organizational or product change because the system would take two years to modify - a clear, if negatively stated, call for an expressive system.

Within Bradford & Bingley's system is a generic products engine, which is built around such a generalized concept of a savings product that it is capable of also serving as a loan or insurance product. Each product is attached, not to a customer, but to a more abstract software construct known as "associate" - defined as a party with which the firm has a relationship. More specific versions of associate include the conventional notion of customer, but also agents and branches (retail outlets). By making all other parts of the software interface with the abstract or common version of these specific entities, a decision to change an agent-based product to a branch-based product has no external impact. Equally, if the firm decided to launch a specialized savings product for its own employees, which might entail special terms or security arrangements, it merely requires a new specialized subclass of associate called "employee," but no change to the product engine. Bradford & Bingley's system also has generic software engines for document creation, for the management of workflow in an administrative process, and for managing selected groups of associates for marketing.

Abstraction is not a new concept in information systems; indeed, it is a basic principle of data modeling, but there have been few attempts to apply this to the functionality (i.e., the code) of systems. Historically, the view has been that code needs to be written to meet the specific needs of an individual application. To the extent that there has been any reuse of existing code, it has been at a very technical level: reusable subroutines for implementing a complex mathematical function or for managing computer resources. Little attention has been given to identifying generic or abstract business functions and implementing these as reusable components.

The advent of object orientation is changing this by replacing the artificial separation of code and data with the more natural concept of self-contained objects. A software object completely models a component of the business domain: it contains the data that represents that component and all the functionality that may change or interact with that data.

Object orientation has a natural fit with expressive systems in several ways: it underpins most sophisticated graphical user interfaces and facilitates the construction of reusable components. However, the greatest significance of object orientation, and the one least understood by most IS professionals, is its ability to support business abstraction. Two principles of object orientation apply here. The first is called inheritance, which facilitates the creation and management of specialized subclasses of objects, as in the relationship between associate and customer. The second is called "polymorphism," in which different objects execute different code in response to the same message. A spreadsheet and a word-processed document can both respond to the message "print," but the way they perform that function will be different. Polymorphism simply reflects the reality that, in business, there are fewer things we want to do than ways we want to do them. Expressive systems should provide the support for the generic things we want to do, and the user's application of that capability implements the particular way it is to be done.

The New IS Department

What kind of IS department will be needed to implement and/or support information systems that are based primarily on the expressive systems model? No single organization that we have encountered has yet completed this transition, but those who have partially moved toward expressive systems have had enough experience that we can piece together a plausible model for the future IS department. Organizations that identify with the potential benefits of expressive systems must recognize that implementing the concept is as much about changing the structure and behavior of the IS department as it is about changing the technology.

The first distinction between this future model and the IS department of today is in the realm of values, attitudes, and beliefs. Take the widespread belief that "If only we could get the users to specify exactly what it is they require, then our problems would be solved." The concept of expressive systems, unlike iterative development, is not based on the notion that users have difficulty expressing their exact requirements; it is based on the realization that increasingly users cannot state their requirements completely because they themselves cannot know all the business conditions and events they will be facing.

A second distinction concerns the role of systems themselves. In the future model, the role of systems is primarily to facilitate change. This means that the IS department must anticipate business change. It does not mean that IS must predict business change; rather it means that it must build systems that are resilient to future change through the use of abstraction, componentization, and rewirable infrastructure.

Thirdly, IS professionals must change their current beliefs about the difference between developers and users. The distinction has been historically valid because the user and developer dealt with different views of the same system. The developer's view comprised lines of programming code, data structures, and operating system calls, which together form a high-level representation of the computers resources. The user's view comprised menus of commands, forms to be filled, queries and reports, which together form a representation of the specific business operations being supported. It is the translation between these two representations that makes conventional systems development such a time-consuming, arduous process. Expressive systems resolve the problem by having the developer and the user share the same representation - a representation of the natural components and structure of the problem domain rather than a specific solution to it.[11]

Consider, for example, a spreadsheet. Its success lies in the fact that its constructs (tables, cells, and formulas) are a very natural representation of the structure of financial modeling problems and are suited to both very simple and very complex applications. If you have used a spreadsheet, you have almost certainly developed an application; moreover, between developing the spreadsheet and using the resulting application, there is no switch in the representation used. There is a clear distinction between the authors of the spreadsheet package (Microsoft or Lotus, say) and the user/developers, but this is not the same relationship as between conventional systems developers and users.

It is a curious fact that while many IS departments acknowledge the very sophisticated spreadsheet applications within their businesses, few provide any real support for spreadsheet development. Indeed, there is often an underlying cynicism with regard to end-user development in general. Professional developers are fond of quoting surveys demonstrating that 70 percent of all spreadsheets contain some kind of error but are less keen to help reduce those errors.

In the expressive systems era, professional developers still have roles to play, but those roles will change. Some will be deployed in the creation, management, and continuous enhancement of the infrastructure. This, we believe, will divide into three processes: "Business model maintenance" will be concerned exclusively with the uncommitted software model. Its developers will be high-caliber abstract thinkers, and a key issue will be the communication of the knowledge of this model to those applying it. The second process is concerned with the technologies that support the business model. One of the keys will be ensuring that significant new technologies are introduced to the system in the form of

generic infrastructure services, rather than as specific applications. The final process we might call technology services, which most closely resembles the IS operations function today, except that it will be concerned with monitoring and improving performance at the level of individual software components rather than just of whole systems.

The professional developers' other role will be to act as mentors within the application process.(12) For example, in the sales and marketing department of Clorox Company, developers have completely adopted this role. In 1985, Clorox installed one of the earliest, truly expressive data retrieval and manipulation packages, Metaphor (which was also an early example of an **object-oriented** system). In Metaphor, users write modules or capsules and then visually wire the capsules together to generate the reports or screens they require. Almost 150 people in the sales and marketing department use Metaphor intensively, and they are supported by between one and three systems professionals, as needs vary. As part of their mentoring role, the professionals look for commonality in capsules written by different users. The professionals rewrite those versions into a standard, robust, more flexible capsule and offer it around to all the users (who themselves often share capsules with each other via e-mail). The notion of improving user-developed systems would be anathema to many systems professionals. But Clorox has found that the net ratio of functionality created to professional systems input exceeds every other application of information systems in the company - which more or less sums up the case for expressive systems.

References

1. E. Dyson, ed., "An Explicit Look at Explicitness," Release 1.0, October 1993, pp.1-19.

2. B. Nardi, A Small Matter of Programming (Cambridge, Massachusetts: MIT Press, 1993).

3. M. Treacy and F. Wiersema, "Customer Intimacy and Other Value Disciplines," Harvard Business Review, January-February 1993, pp. 84-93.

4 T. Malone, J. Yates, and R. Benjamin, "Electronic Markets and Electronic Hierarchies," Communications of the ACM 30 (1987): 484-497.

5. H. Mintzberg, Mintzberg an Management (New York: Free Press, 1989), p. 243.

6. S. Haeckel and R. Nolan, "Managing by Wire," Harvard Business Review, September-October 1993, pp. 122-132.

7. J. Rockart and D. De Long, Executive Support Systems (Homewood, Illinois: Dow Jones-Irwin, 1988).

8. J. Martin, Rapid Application Development (New York: Macmillan, 1991).

9. R. Pawson et al., Building the New Information Infrastructure (London: CSC Foundation, 1993).

10. R. Morison, The New I/S Agenda (Cambridge, Massachusetts: CSC Research and Advisory Services, 1994).

11. J. Tibbets, "Object Orientation and Transaction Processing, Where Do They Meet?" (Phoenix, Arizona: OOPSLA 91, Sixth Annual Conference, addendum to the proceedings, 6-11 October 1991), pp. 3-15.

12. B. Nardi and J. Miller, "Twinkling Lights and Nested Loops: Distributed Problem Solving and Spreadsheet Development," International Journal of Man-Machine Studies 34 (1991): 161-184.

Richard Pawson is European director of research for CSC Research and Advisory Services. Jean-Louis Bravard is a managing director at J.P. Morgan & Company. Lorette Cameron is a vice president at J.P. Morgan.

INDUSTRY CODES/NAMES: BUS Business, General
DESCRIPTORS: Information systems--Evaluation
FILE SEGMENT: MC File 75

07565926 SUPPLIER NUMBER: 15930887 (THIS IS THE FULL TEXT)
D&B Software Announces Significant New Releases of SmartStream Series.
Business Wire, p11141687
Nov 14, 1994
LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT
WORD COUNT: 2119 LINE COUNT: 00198

TEXT:

ATLANTA--(BUSINESS WIRE)--Nov. 14, 1994-- Dun & Bradstreet Software today announced SmartStream 3.0, a major new release of its SmartStream enterprise suite of integrated workflow-enabled client/server tools and applications which deliver significant functionality and enhancements.

The announcements today of the newly available StreamBuilder, Manufacturing Stream, Distribution Stream, and the major releases of Financial Stream, HR Stream and SmartStream Decision Support, bring to a total of eight new products, support of four new platforms and two languages, introduced this year.

The SmartStream Series is a Microsoft Windows-based client/server-based enterprise solution comprised of integrated business applications, comprehensive decision support tools and a robust workflow-enabled platform that allows organizations to re-engineer their business processes to maximize competitiveness in their industry.

SmartStream 3.0 also includes an improved graphical interface, unparalleled information delivery, new international financial management and reporting capabilities, purchasing and allocations modules, and a business process orientation based on workflow and agent technologies. These improvements address enterprise-wide information management requirements of finance, sales and distribution professionals.

"With SmartStream, D&B Software's integrated client/server applications make a great leap forward to robust, enterprise-wide application functionality," said R. Douglas MacIntyre, president and chief executive officer. "Our client/server applications are designed to address global requirements of the most discerning customers."

SmartStream 3.0 Series includes:

- StreamBuilder -- an application development and customizing tool kit that offers business and systems analysts, programmers and others a proven alternative to traditional development methods.

- Manufacturing Stream 3.0 -- integrated manufacturing applications for discrete and repetitive environments. It supports flexible manufacturing strategies that exploit just-in-time production principles and Kanban inventory management.

- Distribution Stream 3.0 -- integrated distribution applications that support flexible distribution strategies that embrace total supply chain management.

- Financial Stream 3.0 -- a new version of D&B Softwares financial management system that is available in English and now in French. It also includes the availability of purchasing and allocations modules and major functionality enhancements in asset management, accounts payable, accounts receivable, and international reporting.

- SmartStream Decision Support 3.0 -- D&B Softwares premier analysis and reporting client/server tool includes improvements to the applications SQL engine, as well as enhanced management report development and distribution capabilities. Among the end-user benefits offered in SmartStream Decision Support 3.0 include a major improvement in reporting throughput capacity which delivers a significant performance improvement in the delivery of management reports.

- SmartStream Budget 3.0 -- a new version of D&B Software's integrated budgeting and planning application that includes additional functionality in the areas of creating budgets, viewing and printing reports and integrating with other modules.

- HR Stream 3.0 -- D&B Software's integrated human resources applications has added enhancements in the areas of translation, security and integration with host-based systems.

"Release 3.0 of our SmartStream architecture provides a powerful information reporting and delivery capability that leverages our continued

investment in workflow technologies and provides a truly open architecture upon which our customers can build their enterprise-wide business solutions," said MacIntyre.

StreamBuilder (see separate announcement for more details)

StreamBuilder provides an **object-oriented,** Microsoft Windows-based environment that lets a user create add-ons and customer-specific applications with the same look and feel as SmartStream applications. With StreamBuilder users can also customize their current SmartStream applications and integrate non-D&B Software applications within SmartStream workflows.

Manufacturing Stream and Distribution Stream (see separate announcement for more details)

Components in this release include: Product Definition; Manufacturing Planning; Inventory Management; Dock-to-Stock/Receiving; Purchasing and Order Management. Product Definition defines and maintains all items and creates and maintains the product structures used by an enterprise. Manufacturing Planning analyzes the current status of each item in inventory, site-by site, and creates replenishment schedules. Purchasing supports the entire procurement cycle from the requisitioner's desk to the placement of purchase orders with a vendor. Dock-to-Stock, Receiving, and Order Management handles the complete management cycle of material movement, from inbound receipt, to customer order management, shipment and invoicing of orders.

Financial Stream 3.0 Enhancements:

Financial Steam 3.0 is an enterprise-wide system providing a core of integrated financial applications that enable cross-functional execution and monitoring of critical financial information and business processes and provide quick feedback for proactive decision making. Financial Stream includes areas traditionally associated with general ledger, accounts payable, accounts receivable and fixed assets. Unlike traditional and competitive financial applications, Financial Stream organizes business objects around activities such as journal processing, payment request, asset management and currency translation. This approach gives customers the ability to easily individualize its business processes.

Purchasing

SmartStream Purchasing enables customers to leverage volume purchasing by allowing them to consolidate purchasing for multiple sites, with multiple delivery dates and multiple ship-to locations, and with multiple accounting distributions to a single purchase user. SmartStream Purchasing includes distributed processing, logical or physical, as well as items, vendors, vendor items, requisitions, purchase orders, blanket agreements and EDI capability.

SmartStream Purchasing was developed to take advantage of workflow and intelligent agent technology, providing increased efficiencies for end users. For example, SmartStream Purchasing can automatically detect a receiving quantity exception and delivers it via mail distribution to the buyer responsible for that particular order so it can be resolved. Following order resolution, the agent sends notification to the receiving department for completion. This eliminates the time spent by users on non-strategic tasks. SmartStream Purchasing is a stand-alone module that is fully integrated with Financial Stream's Financial Records and Payables modules.

Allocations Module

SmartStream Allocations, an allocations module that enables users to more effectively allocate indirect line-of-business or product-line costs and revenues is now available with Financial Stream 3.0. This provides an organization with a better understanding of the true profitability of a unit, division or product category. SmartStream Allocations is fully integrated with Financial Stream's Financial Records module and SmartStream Budget. It also can be used as a stand-alone allocation tool or in conjunction with SmartStream Decision Support.

Matching

Integration between the Purchasing and Payables applications allows a user to electronically match purchase order, receipt and invoice information at the purchase order line schedule level. Selected invoices

are compared to the purchase orders and the related receiving documents to establish any exceptions. If exceptions are associated with the invoice, the information necessary to research the exception is viewed on-line. Exception errors can then be corrected by drilling down to the original source system.

SmartStream's workflow facilitates efficient customized exception routing. Since different types of matching exceptions are typically resolved by people in different functional areas, To Do messages with the accompanying matching exception information are routed to the appropriate person based on the type of exception. For example, if an exception occurs when an invoice tries to match against a canceled purchase order, the workflow could be set up to specifically route the error to the user who can investigate and correct the error.

Payables Drafts

Financial Stream 3.0 delivers drafts as an additional payment method available for European customers. Drafts are a **financial instrument** for the vendor that can be discounted or used as collateral for lines of credit. A draft can be created by the vendor issuing the draft, or Bill of Exchange, from his receivables system, or the customer issuing the draft, or Promissory Note, from her payables system.

Combination Validation

Financial Stream 3.0 also includes a "combination checking" function which simplifies the process of adding new account, cost center or product line information. "Combination checking" allows users to easily establish a table of permitted or not allowed combinations of key field information, significantly reducing time spent maintaining the system and posting errors. For example, an oil and gas exploration company, with more than 100,000 wells, has a reporting structure that requires it to add and validate information about multiple locations simultaneously. Since only accounting key combinations needed for posting are created, combination checking can also reduce file sizes.

Asset Depreciation

Financial Stream 3.0 includes enhanced international features to support companies with multi-location Asset Depreciation Range (ADR) requirements. These include the ability to perform ADR accounting and asset retirement reversal and management of changes in scheduled processing, including simplification of the bulk copy procedure and the addition of ADR fields.

SmartStream Budget Enhancements

SmartStream Budget is a fully integrated, client/server budget application which provides an entire enterprise with the ability to shorten the time required to create budgets, automates budget data distribution, enables implementation of a consistent budgeting methodology and provides superior analysis and reporting capability. SmartStream Budget 3.0 has added functionality which includes:

- Integrating with SmartStream Allocations.
- Creating Rolling Budgets, which allows users to continuously budget 12 months into the future.
- Allowing users to prepare budgets more easily through an enhanced workbench windows environment.
- Enabling users to create, view and print reports directly from the workbench.

HR Stream 3.0 Enhancements

HR Stream provides information on fundamental human resources functions from hiring through termination on both a local and global level. These business functions include job and position management, recruitment, employment compensation, training and government compliance. The system integrates with host-based payroll systems and with D&B Software's SmartStream Series of client/server products.

HR Stream 3.0 has added functionality which includes:

- Enhanced data security functionality for viewing of sensitive personnel data such as salary information.
- Compliance with international standards which makes it easier to convert HR Stream to other languages.
- Enhanced interface to the host for activities like payroll

processing.

SmartStream Decision Support 3.0 Enhancements:

SmartStream Decision Support 3.0 offers new capabilities designed to allow faster information access and easier and broader information distribution.

Improved Data Delivery

Using SmartStream Decision Support 3.0s SmartStream Query & Reporter, customers can create a "distribute by structure" option. This enables customers to distribute common reports to multiple recipients or to deliver reports of varying detail or "views" to multiple sites across the enterprise. For example, a western region sales report listing all sales during the quarter can be sent to the regional sales manager, while streamlined reports listing only particular accounts in that region can be sent to individual sales representatives.

For departments or organizations with requirements to perform analysis and reporting for multiple executives or locations, SmartStream 3.0 now provides the ability to handle multiple management reports in a SmartStream application and simultaneously deliver multiple reports or "information packets" to numerous users simultaneously through any MAPI or VIM-compliant electronic mail system. By enabling users to group reports together by recipient, SmartStream Decision Support is further streamlining the information delivery process.

Pricing and Availability

Financial Stream 3.0, HR Stream 3.0 and SmartStream Decision Support 3.0 is shipping this month. Financial Stream and HR Stream run on Hewlett-Packards HP-UX, Data Generals DG/UX, IBMs RS/6000 AIX, and Sun Microsystem's Solaris. SmartStream Decision Support runs on Hewlett-Packards HP- UX, Data Generals DG/UX, IBMs RS/6000 AIX and OS/2, Sun Microsystem's Solaris, Intel-based Microsoft NT, Digital's Alpha OSF/1 and ICL DRS 6000. Pricing for Financial Stream starts at $250,000. Pricing for SmartStream Decision Support and HR Stream each starts at $100,000. Pricing for StreamBuilder starts at $20,000 per module. Manufacturing Stream and Distribution Stream are priced from $300,000. Pricing depends on system configuration and number of users.

D&B Software, with more than 10,000 customer sites in over 60 countries, is a company of The Dun & Bradstreet Corporation. D&B Software, with more than 2,000 employees, provides a broad range of business software products, tools and services, including decision support tools, financial, human resource, manufacturing and distribution applications.

SmartStream and Financial Stream are registered trademarks and StreamBuilder, SmartStream Budget, SmartStream Decision Support, HR Stream, Manufacturing Stream and Distribution Stream are trademarks of Dun & Bradstreet Software Services, Inc. Throughout this release, other software and hardware products are mentioned by name. In most, if not all cases, these product names are claimed as trademarks by the companies that manufacture the products. It is not our intention to claim these names or trademarks as our own.

CONTACT: D&B Software, Atlanta
Lorretta Gasper, 404/239-3658
or
Copithorne & Bellows, Boston
Tim Hurley/Dave Copithorne 617/252-0606

COMPANY NAMES: Dun and Bradstreet Software Services Inc.--Product introduction
INDUSTRY CODES/NAMES: BUS Business, General
DESCRIPTORS: Computer software industry--Product introduction
PRODUCT/INDUSTRY NAMES: 7372311 (Financial, Acctg Software (ex Micro))
SIC CODES: 7372 Prepackaged software
FILE SEGMENT: NW File 649

07565784 SUPPLIER NUMBER: 15928543 (THIS IS THE FULL TEXT)
D&B Software Announces Significant New Releases of SmartStream Series.
Business Wire, p11151051
Nov 15, 1994
LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT
WORD COUNT: 2119 LINE COUNT: 00198

TEXT:

ATLANTA--(BUSINESS WIRE)--Nov. 15, 1994--Dun & Bradstreet Software today announced SmartStream 3.0, a major new release of its SmartStream enterprise suite of integrated workflow-enabled client/server tools and applications which deliver significant functionality and enhancements.

The announcements today of the newly available StreamBuilder, Manufacturing Stream, Distribution Stream, and the major releases of Financial Stream, HR Stream and SmartStream Decision Support, bring to a total of eight new products, support of four new platforms and two languages, introduced this year.

The SmartStream Series is a Microsoft Windows-based client/server-based enterprise solution comprised of integrated business applications, comprehensive decision support tools and a robust workflow-enabled platform that allows organizations to re-engineer their business processes to maximize competitiveness in their industry.

SmartStream 3.0 also includes an improved graphical interface, unparalleled information delivery, new international financial management and reporting capabilities, purchasing and allocations modules, and a business process orientation based on workflow and agent technologies. These improvements address enterprise-wide information management requirements of finance, sales and distribution professionals.

"With SmartStream, D&B Software's integrated client/server applications make a great leap forward to robust, enterprise-wide application functionality," said R. Douglas MacIntyre, president and chief executive officer. "Our client/server applications are designed to address global requirements of the most discerning customers."

SmartStream 3.0 Series includes:

- StreamBuilder -- an application development and customizing tool kit that offers business and systems analysts, programmers and others a proven alternative to traditional development methods.

- Manufacturing Stream 3.0 -- integrated manufacturing applications for discrete and repetitive environments. It supports flexible manufacturing strategies that exploit just-in-time production principles and Kanban inventory management.

- Distribution Stream 3.0 -- integrated distribution applications that support flexible distribution strategies that embrace total supply chain management.

- Financial Stream 3.0 -- a new version of D&B Softwares financial management system that is available in English and now in French. It also includes the availability of purchasing and allocations modules and major functionality enhancements in asset management, accounts payable, accounts receivable, and international reporting.

- SmartStream Decision Support 3.0 -- D&B Softwares premier analysis and reporting client/server tool includes improvements to the applications SQL engine, as well as enhanced management report development and distribution capabilities. Among the end-user benefits offered in SmartStream Decision Support 3.0 include a major improvement in reporting throughput capacity which delivers a significant performance improvement in the delivery of management reports.

- SmartStream Budget 3.0 -- a new version of D&B Software's integrated budgeting and planning application that includes additional functionality in the areas of creating budgets, viewing and printing reports and integrating with other modules.

- HR Stream 3.0 -- D&B Software's integrated human resources applications has added enhancements in the areas of translation, security and integration with host-based systems.

"Release 3.0 of our SmartStream architecture provides a powerful information reporting and delivery capability that leverages our continued investment in workflow technologies and provides a truly open architecture upon which our customers can build their enterprise-wide business solutions," said MacIntyre.

StreamBuilder (see separate announcement for more details)

StreamBuilder provides an **object-oriented,** Microsoft Windows-based environment that lets a user create add-ons and customer-specific applications with the same look and feel as SmartStream applications. With StreamBuilder users can also customize their current SmartStream applications and integrate non-D&B Software applications within SmartStream workflows.

Manufacturing Stream and Distribution Stream (see separate announcement for more details)

Components in this release include: Product Definition; Manufacturing Planning; Inventory Management; Dock-to-Stock/Receiving; Purchasing and Order Management. Product Definition defines and maintains all items and creates and maintains the product structures used by an enterprise. Manufacturing Planning analyzes the current status of each item in inventory, site-by site, and creates replenishment schedules. Purchasing supports the entire procurement cycle from the requisitioner's desk to the placement of purchase orders with a vendor. Dock-to-Stock, Receiving, and Order Management handles the complete management cycle of material movement, from inbound receipt, to customer order management, shipment and invoicing of orders.

Financial Stream 3.0 Enhancements:

Financial Steam 3.0 is an enterprise-wide system providing a core of integrated financial applications that enable cross-functional execution and monitoring of critical financial information and business processes and provide quick feedback for proactive decision making. Financial Stream includes areas traditionally associated with general ledger, accounts payable, accounts receivable and fixed assets. Unlike traditional and competitive financial applications, Financial Stream organizes business objects around activities such as journal processing, payment request, asset management and currency translation. This approach gives customers the ability to easily individualize its business processes.

Purchasing

SmartStream Purchasing enables customers to leverage volume purchasing by allowing them to consolidate purchasing for multiple sites, with multiple delivery dates and multiple ship-to locations, and with multiple accounting distributions to a single purchase user. SmartStream Purchasing includes distributed processing, logical or physical, as well as items, vendors, vendor items, requisitions, purchase orders, blanket agreements and EDI capability.

SmartStream Purchasing was developed to take advantage of workflow and intelligent agent technology, providing increased efficiencies for end users. For example, SmartStream Purchasing can automatically detect a receiving quantity exception and delivers it via mail distribution to the buyer responsible for that particular order so it can be resolved. Following order resolution, the agent sends notification to the receiving department for completion. This eliminates the time spent by users on non-strategic tasks. SmartStream Purchasing is a stand-alone module that is fully integrated with Financial Stream's Financial Records and Payables modules.

Allocations Module

SmartStream Allocations, an allocations module that enables users to more effectively allocate indirect line-of-business or product-line costs and revenues is now available with Financial Stream 3.0. This provides an organization with a better understanding of the true profitability of a unit, division or product category. SmartStream Allocations is fully integrated with Financial Stream's Financial Records module and SmartStream Budget. It also can be used as a stand-alone allocation tool or in conjunction with SmartStream Decision Support.

Matching

Integration between the Purchasing and Payables applications allows a

user to electronically match purchase order, receipt and invoice information at the purchase order line schedule level. Selected invoices are compared to the purchase orders and the related receiving documents to establish any exceptions. If exceptions are associated with the invoice, the information necessary to research the exception is viewed on-line. Exception errors can then be corrected by drilling down to the original source system.

SmartStream's workflow facilitates efficient customized exception routing. Since different types of matching exceptions are typically resolved by people in different functional areas, To Do messages with the accompanying matching exception information are routed to the appropriate person based on the type of exception. For example, if an exception occurs when an invoice tries to match against a canceled purchase order, the workflow could be set up to specifically route the error to the user who can investigate and correct the error.

Payables Drafts

Financial Stream 3.0 delivers drafts as an additional payment method available for European customers. Drafts are a **financial instrument** for the vendor that can be discounted or used as collateral for lines of credit. A draft can be created by the vendor issuing the draft, or Bill of Exchange, from his receivables system, or the customer issuing the draft, or Promissory Note, from her payables system.

Combination Validation

Financial Stream 3.0 also includes a "combination checking" function which simplifies the process of adding new account, cost center or product line information. "Combination checking" allows users to easily establish a table of permitted or not allowed combinations of key field information, significantly reducing time spent maintaining the system and posting errors. For example, an oil and gas exploration company, with more than 100,000 wells, has a reporting structure that requires it to add and validate information about multiple locations simultaneously. Since only accounting key combinations needed for posting are created, combination checking can also reduce file sizes.

Asset Depreciation

Financial Stream 3.0 includes enhanced international features to support companies with multi-location Asset Depreciation Range (ADR) requirements. These include the ability to perform ADR accounting and asset retirement reversal and management of changes in scheduled processing, including simplification of the bulk copy procedure and the addition of ADR fields.

SmartStream Budget Enhancements

SmartStream Budget is a fully integrated, client/server budget application which provides an entire enterprise with the ability to shorten the time required to create budgets, automates budget data distribution, enables implementation of a consistent budgeting methodology and provides superior analysis and reporting capability. SmartStream Budget 3.0 has added functionality which includes:

- Integrating with SmartStream Allocations.
- Creating Rolling Budgets, which allows users to continuously budget 12 months into the future.
- Allowing users to prepare budgets more easily through an enhanced workbench windows environment.
- Enabling users to create, view and print reports directly from the workbench.

HR Stream 3.0 Enhancements

HR Stream provides information on fundamental human resources functions from hiring through termination on both a local and global level. These business functions include job and position management, recruitment, employment compensation, training and government compliance. The system integrates with host-based payroll systems and with D&B Software's SmartStream Series of client/server products.

HR Stream 3.0 has added functionality which includes:

- Enhanced data security functionality for viewing of sensitive personnel data such as salary information.
- Compliance with international standards which makes it easier to

convert HR Stream to other languages.

- Enhanced interface to the host for activities like payroll processing.

SmartStream Decision Support 3.0 Enhancements:

SmartStream Decision Support 3.0 offers new capabilities designed to allow faster information access and easier and broader information distribution.

Improved Data Delivery

Using SmartStream Decision Support 3.0s SmartStream Query & Reporter, customers can create a "distribute by structure" option. This enables customers to distribute common reports to multiple recipients or to deliver reports of varying detail or "views" to multiple sites across the enterprise. For example, a western region sales report listing all sales during the quarter can be sent to the regional sales manager, while streamlined reports listing only particular accounts in that region can be sent to individual sales representatives.

For departments or organizations with requirements to perform analysis and reporting for multiple executives or locations, SmartStream 3.0 now provides the ability to handle multiple management reports in a SmartStream application and simultaneously deliver multiple reports or "information packets" to numerous users simultaneously through any MAPI or VIM-compliant electronic mail system. By enabling users to group reports together by recipient, SmartStream Decision Support is further streamlining the information delivery process.

Pricing and Availability

Financial Stream 3.0, HR Stream 3.0 and SmartStream Decision Support 3.0 is shipping this month. Financial Stream and HR Stream run on Hewlett-Packards HP-UX, Data Generals DG/UX, IBMs RS/6000 AIX, and Sun Microsystem's Solaris. SmartStream Decision Support runs on Hewlett-Packards HP- UX, Data Generals DG/UX, IBMs RS/6000 AIX and OS/2, Sun Microsystem's Solaris, Intel-based Microsoft NT, Digital's Alpha OSF/1 and ICL DRS 6000. Pricing for Financial Stream starts at $250,000. Pricing for SmartStream Decision Support and HR Stream each starts at $100,000. Pricing for StreamBuilder starts at $20,000 per module. Manufacturing Stream and Distribution Stream are priced from $300,000. Pricing depends on system configuration and number of users.

D&B Software, with more than 10,000 customer sites in over 60 countries, is a company of The Dun & Bradstreet Corporation. D&B Software, with more than 2,000 employees, provides a broad range of business software products, tools and services, including decision support tools, financial, human resource, manufacturing and distribution applications.

SmartStream and Financial Stream are registered trademarks and StreamBuilder, SmartStream Budget, SmartStream Decision Support, HR Stream, Manufacturing Stream and Distribution Stream are trademarks of Dun & Bradstreet Software Services, Inc. Throughout this release, other software and hardware products are mentioned by name. In most, if not all cases, these product names are claimed as trademarks by the companies that manufacture the products. It is not our intention to claim these names or trademarks as our own.

CONTACT: D&B Software, Atlanta
Lorretta Gasper, 404/239-3658
or
Copithorne & Bellows, Boston
Tim Hurley/Dave Copithorne 617/252-0606

COMPANY NAMES: Dun and Bradstreet Software Services Inc.--Product introduction
INDUSTRY CODES/NAMES: BUS Business, General
DESCRIPTORS: Computer software industry--Product introduction
PRODUCT/INDUSTRY NAMES: 7372419 (Business Software (Micro))
SIC CODES: 7372 Prepackaged software
FILE SEGMENT: NW File 649

06702097 SUPPLIER NUMBER: 14330777 (THIS IS THE FULL TEXT)
Prometheus unplugged. (wireless communication is poised to revolutionize financial trading) (Forbes ASAP: A Technology Supplement)(Quick Studies)
Young, Jeffrey
Forbes, v152, n6, pS149(2)
Sept 13, 1993
ISSN: 0015-6914 LANGUAGE: ENGLISH RECORD TYPE: FULLTEXT; ABSTRACT
WORD COUNT: 1334 LINE COUNT: 00101

ABSTRACT: Swiss Bank Corp is aggressively pursuing the application of wireless computing and communications technology despite its history as one of the most conservative of Swiss banks. The company has supplemented its array of high-end workstations with various wireless communications devices such as radio pagers and palmtop computers. Dwight Koop, one of the bank's information technology executives, hopes eventually to have the bank's commodity traders feed prices and sales information by handheld computers that they could carry on the floor of the Chicago Board of Trade. Swiss Bank is also in the process of completing its acquisition of O'Connor Partnerships, a Chicago trading operation that uses sophisticated software to manage trading in financial derivatives, such as options and index instruments.
TEXT:

Four floors above the Chicago Board of Trade's commodities futures pits lies trading's real future.

DWIGHT KOOP WORKS for one of the world's largest, and most conservative, Swiss banks. As he attends a meeting in an elegant conference room, the beep from the pair of palm-sized black boxes on the table in front of him is barely audible. Nestled in a leather carrying case and held in place by Velcro fasteners, the Hewlett Packard 100LX, a top-of-the-line programmable calculator-cum-palmtop-computer, is tethered to an Ericsson GE Mobidem wireless modem whose flexible plastic antenna is waving in the breeze from the air conditioning. Both devices are battery-powered. A symbol atop the modem's LCD screen flashes on and off every few seconds, indicating that something is coming in, or going out.

That's not the only thing going on.

In the meeting room, the voice of an executive in Switzerland booms out of the speakerphone. Beyond the glass wall, a mezzanine gallery circles a basketball-court-sized three-level space filled with pods of desks, stacks of computer monitors and the requisite wall of moving ticker symbols, stock prices and news feeds. This is the trading floor of the Swiss Bank Corp. in Chicago, four floors above the chaos of the Chicago Board of Trade's commodities futures pits. Here, in contrast to the frenzy of colored jackets and hand gestures in the pits, all the action happens on Next, Sun Microsystems, Hewlett Packard, Symbolics and other workstations--as well as on a growing number of handheld, wireless devices of all kinds.

This afternoon the group in the meeting room is discussing new software developments that Swiss Bank is planning. But Dwight Koop, 45 years old and executive director of information technology for Swiss Bank Corp., has other things on his mind. A few taps on the tiny keypad move him through a list of 15 E-mail messages that have landed in his wireless mail-box with the latest beep. The HP 100 and modem are small and unobtrusive enough that he can use them without disrupting the meeting. He stops at one message, pulls it up on the screen and squints at it through his glasses. A faintly worried look passes over his face. A few more taps and a reply shoots out into the ether. Simultaneously, his SkyTel Sky-Word beeper starts vibrating. Koop checks it, then punches his watch, a Dick Tracy-like digital artifact that not only tells the time but also is a paging device. He stands up, closes his equipment, mumbles something about "putting out a fire" under his breath so the telecommuting executive can't quite hear him over the speakerphone, and breaks away from the meeting.

Koop loves the gadgets. But he's also charged with integrating them into Swiss Bank's work culture, and that, he says, won't be easy: "It's not all ready for prime time yet. The service's interface and interaction model is troubling. For someone who has spent years messing with computers, it can be navigated. Far too often I have to fight it [the system] to make it work."

Koops says wireless trading will "languish" until somebody figures out all the pieces and integrates them. Today Swiss Bank sends its traders onto the floors of the exchanges with print-outs from its internal computer systems. The bank would like to someday equip its traders with handheld machines. "It won't be anytime soon," Koop says. "Trading is a contact sport."

Koop's job involves keeping the bank's global trading systems up and running. These are not just any trading systems. Over the past few years, Swiss Bank has acquired the majority of Chicago's O'Connor Partnerships (the final portion of the acquisition is awaiting SEC approval). O'Connor is one of the most highly secretive, technically sophisticated and profitable of the world's trading operations dealing in financial derivatives--products like options and index instruments that are derived from underlying traded securities such as stocks, bonds and currencies. A leader in the application of mathematical algorithms to options, currency and **financial instrument** trading, O'Connor was once known for trying to avoid all publicity. It once shredded the packaging materials for the workstations it bought and required all employees to sign extensive secrecy agreements. Even today the closest a visitor can get to an O'Connor computer screen is the gallery, a good 30 feet from the trading floor.

Creating new bundles of instruments and options to sell in response to customer requests--instantly--is a growth opportunity that savvy traders have been eager to exploit. All of this is much better handled by computers, which O'Connor realized in the mid-1980s when it led the financial community in moving first to Sun machines and later to more sophisticated workstations. But the new trader's edge is literally up in the air.

WALKING AROUND--GLOBALLY

If any trading company figures out wireless, it probably will be Swiss. Swiss Bank likes to push the technolo-envelope, for example, buying 500 Next machines a few years ago because it believed in the future of **object-oriented** development systems. "We have no idea if Next will make it in the long run," says Koop's boss, Craig Heimark, Swiss Bank's managing director of technology. "But someone will succeed with objects. And we'll already have a wealth of experience programming in this kind of environment."

Koop enjoys explaining that he has signed 425 nondisclosure agreements with technology companies. "When companies ask if they can qualify us as a beta site," he says, "I explain that they don't seem to understand. We qualify them." This is real money being handled--security is a big issue. "Dial-in modems simply don't exist in our world," he adds quietly.

For all of O'Connor's history of secrecy, Swiss Bank is talking about moving its proprietary risk-management trading systems to customer sites. The idea is for the company to let customers manipulate versions of its proprietary analysis tools, then sell them the financial products that meet their needs. Enter wireless. "We want our managers to work by walking around--globally," Koop explains. "But we sure don't want these top-level people fumbling for the phone jack and negotiating with the PBX operator to get a dial-out line so they can check the day's market."

For all his complaining, Koop is still certain that wireless will be a key communications component over the next few years. His own love of the toys convinces him.

On another day Koop is sitting in a hotel in Sausalito, at a window over-looking San Francisco Bay and the city skyline. His computer and modem are open in front of him, and the radio signal is strong. The indicator is flashing furiously. His attention is intermittently pulled to the screen. It is more than a little disconcerting to try to talk with him. We may have to learn a new social dynamic in the era of ubiquitous wireless machinery.

Suddenly he chuckles, and pushes the unit over to display a message he's just received. "A bunch of us have an unofficial contest to send a message from the most unusual location," he says. "Have you ever heard of this place?" The message is tough to see in the glare, but when the tiny LCD display is positioned correctly, it reads: "Do I win the contest? I'm sitting at the bar of the Jaguar Club in Atlanta." (The Jaguar Club is one of the more infamous topless bars in the South.) Ah, brave new world. As it turned out, that message didn't win. The winning one was sent from atop Elvis' grave at Graceland.

SPECIAL FEATURES: illustration; photograph
COMPANY NAMES: Swiss Bank Corp. (Switzerland)--Communication systems
INDUSTRY CODES/NAMES: BUS Business, General
DESCRIPTORS: Wireless local area networks (Computer networks)--Usage; Financial markets--Communication systems
FILE SEGMENT: MI File 47

4/9/33 (Item 1 from file: 2)
DIALOG(R)File 2:INSPEC

03985566 INSPEC Abstract Number: C91065251
Title: An intelligent assistant for financial hedging
Author(s): Benaroch, M.; Dhar, V.
Author Affiliation: Dept. of Inf. Syst., New York Univ., NY, USA
Conference Title: Proceedings. Seventh IEEE Conference on Artificial Intelligence Applications (Cat. No.91CH2967-8) p.168-74
Publisher: IEEE Comput. Soc. Press, Los Alamitos, CA, USA
Publication Date: 1991 Country of Publication: USA 2 vol. (xix+460+504) pp.
ISBN: 0 8186 2135 4
U.S. Copyright Clearance Center Code: CH2967-8/91/0000-0168$01.00
Conference Sponsor: IEEE
Conference Date: 24-28 Feb. 1991 Conference Location: Miami Beach, FL, USA
Language: English Document Type: Conference Paper (PA)
Treatment: Practical (P)
Abstract: The authors describe the knowledge representation used to develop a prototype expert system for financial hedging. This representation captures the deep domain knowledge that experts use to reason about hedging decisions. It allows for reasoning qualitatively based on first principles using the fundamental quantitative valuation models that characterize each **financial instrument**. It also uses **object-oriented** concepts and inheritance to minimize the effort needed to set up the knowledge base and keep it current. It includes a calculus for derivation of qualitative knowledge of one-dimensional-order, which allows it to solve problems where optimality constraints are qualitative. It is flexible enough to reason in terms of the basic principles of risk assessment. (7 Refs)
Descriptors: expert systems; financial data processing; knowledge representation
Identifiers: knowledge representation; expert system; financial hedging; reasoning; fundamental quantitative valuation models; **object-oriented** concepts; inheritance; calculus; qualitative knowledge; optimality constraints; risk assessment
Class Codes: C7120 (Finance); C6170 (Expert systems)

4/9/34 (Item 1 from file: 625)
DIALOG(R)File 625:American Banker Publications

0154541

Tech Type: Big Banks [redacted]cense Cats' Design Software
American Banker - Ja[redacted]ry 30, 1995; Pg. 14; Vol. 160, [redacted]. 19
SECTION HEADING: Technology/Operations
DATELINE: PALO ALTO, Calif.
ARTICLE TYPE: News
WORD COUNT: 99

BYLINE:
Matt Barthel

TEXT:
Cats Software Inc. announced that several large banking companies have licensed its computer-aided-design financial software, known as Ficad.

The institutions include Bankers Trust New York Corp., Tokai Bank Europe, and Mitsubishi Finance International.

Based on an **object-oriented** approach to programming, Ficad enables traders, risk managers, and product developers to create in a few days applications to monitor any **financial instrument**.

"Computer-aided design tools have been successfully applied in every other major industry over the past decade," said Rod. A Beckstrom, chief executive of Cats. This software "brings the power of (computer-aided design) to finance."

COMPANY NAMES (DIALOG GENERATED): Bankers Trust ; Cats Software Inc ; Mitsubishi Finance International ; New York Corp ; Tokai Bank